NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

OPEN SOURCE TOOLS: APPLICATIONS AND IMPLICATIONS

by

Michael D. Stull

December 2000

Thesis Advisor: John Arquilla
Co-Advisor: Timothy Shimeall

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 1999	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Open Source Tools: Applications and Implications			**5. FUNDING NUMBERS**
6. AUTHOR(S) LT Michael D. Stull			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT ***(maximum 200 words)***

The Internet provides users with unparalleled access to a wide variety of open source tools. Some of the tools may be used in conjunction with others or by themselves, often with great disruptive effect on a target. The rapid pace of discovered vulnerabilities in computer systems, along with the cooperation of expert programmers, has given users access to tools that lower the "entry costs" for conducting sophisticated attacks. Internet security is dependent upon reacting effectively to continually changing modes of attack, and is therefore almost always a step behind, in an action-reaction process.

The availability of pre-tailored attack codes gives possible enemies an avenue to attack the US anonymously, with only a small investment of resources. However, attackers do still need both tools and the knowledge of how to use them to carry out most attacks. Still, more knowledge of the proper utilization of open source tools is progressively being coded into these open source tools, opening up the ability to conduct attacks to a higher percentage of the Internet population

14. SUBJECT TERMS Internet, Open Source Tools, Computer Vulnerabilities, Information Operations			**15. NUMBER OF PAGES** 112
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

OPEN SOURCE TOOLS: APPLICATIONS AND IMPLICATIONS

Michael D. Stull
Lieutenant, United States Navy
B.S., United States Naval Academy, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN DEFENSE ANALYSIS
(INFORMATION OPERATIONS)

from the

NAVAL POSTGRADUATE SCHOOL
December 2000

Author: Michael D. Stull

Approved by: John Arquilla, Thesis Advisor

Timothy Shimeall, Co-Advisor

Gordon McCormick, Chairman
Special Operations Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Internet provides users with unparalleled access to a wide variety of open source tools. Some of the tools may be used in conjunction with others or by themselves, often with great disruptive effect on a target. The rapid pace of discovered vulnerabilities in computer systems, along with the cooperation of expert programmers, has given users access to tools that lower the "entry costs" for conducting sophisticated attacks. Internet security is dependent upon reacting effectively to continually changing modes of attack, and is therefore almost always a step behind, in an action-reaction process.

The availability of pre-tailored attack codes gives possible enemies an avenue to attack the US anonymously, with only a small investment of resources. However, attackers do still need both tools and the knowledge of how to use them to carry out most attacks. Still, more knowledge of the proper utilization of open source tools is progressively being coded into these open source tools, opening up the ability to conduct attacks to a higher percentage of the Internet population

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I wish to sincerely thank my wife Sarah and my family for their support and motivation. Additionally, I wish to thank Dr. Arquilla for his advice and encouragement for this study.

\

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE

Attackers are continuously exploiting inherent weaknesses in operating systems. Attackers place malicious code on the Internet where access to these "tools" is open to any user. The availability of pre-tailored attack codes gives possible enemies an avenue to attack the US anonymously and with a small amount of investment. This has led to the popular notion that new entrants to network attack no longer need to learn the intricacies; they can stand on the shoulders of programmers who came before them and simply use their code. Huge strides have been made in the complexity and thoroughness of open source tools, resulting in an unknown threat level, necessitating an open source review of the tools.

This thesis will provide an overview of the threatening tools and techniques that are freely available on the Internet. The reader will be exposed with a broad "state of the union" of the capabilities of the attack tools and the consequences of their use. Discussion of the evolution and future trends of attacks will be included.

B. RESEARCH QUESTIONS

1. What is the future of open source tools? i.e. What are the trends and what do they point to for future open source attacks?

2. How can distributed networks be used for different types of attack and defense?

C. METHODOLOGY

This thesis centers around the survey of advanced open source tools and techniques available on the Internet. Selection criteria is based upon the level of success and the sophistication of the tools. Analysis considers how these tools are used in coordination and how they might be used in the future. A trend toward distributed network attack as a major new method is demonstrated by the recent high profile attacks earlier this year: the possibility of employing a similar system by different entities is explored.

D. ORGANIZATION

This thesis is organized into five chapters, from basic theory to the findings of future trends. Following this introduction is a chapter on Vulnerabilities and why tools work in a general way. Included in the chapter is a discussion of network organization and the inherent weaknesses that are being exploited. Chapter III discusses three basic types of attack methods and the necessary actions an attacker would have to take for each one. Chapter IV is a survey of current tools available on the Internet today and three scenarios involving current tools. Chapter V outlines future trends and a few hypothetical scenarios. Included in chapter five is a findings section, summarizing the major points of this study. This study has one Appendix which is the results of a survey to determine 50 of the most popular security tools.

II. VULNERABILITIES

> We build our computers the way we build our cities, over time without a plan, on top of ruins (Ullman, 1998).

This chapter outlines the vulnerabilities that exist in applications and protocols and how they are exploited. It is not the intention of this chapter to give a comprehensive background of all the vulnerabilities and the history behind them, but rather to expose the general reasons why they are "hardwired" into systems, and are still being discovered. The basics of how computers communicate are also discussed to help expose the antecedents of most Internet vulnerabilities. Finally, the last section discusses how coordinated attack tools exploit the combination of weaknesses in the Internet and applications/Operating Systems.

A. VULNERABILITY EXPLOITATION LIFECYCLE

> A few software vulnerabilities account for the majority of successful attacks because attackers are opportunistic...they count on organizations not fixing the problems, and they often attack indiscriminately, by scanning the Internet for vulnerable systems (Sans Institute, 2000).

Vulnerability is defined as a weakness that an attacker exploits to conduct unauthorized actions (Howard, 1997, p. 5). Vulnerabilities might be incorporated into the code of an application or the organization of a system: they could be simple oversights, intentionally coded/designed weaknesses, or just poorly written code. When a weakness is discovered, exploitation follows a common pattern.

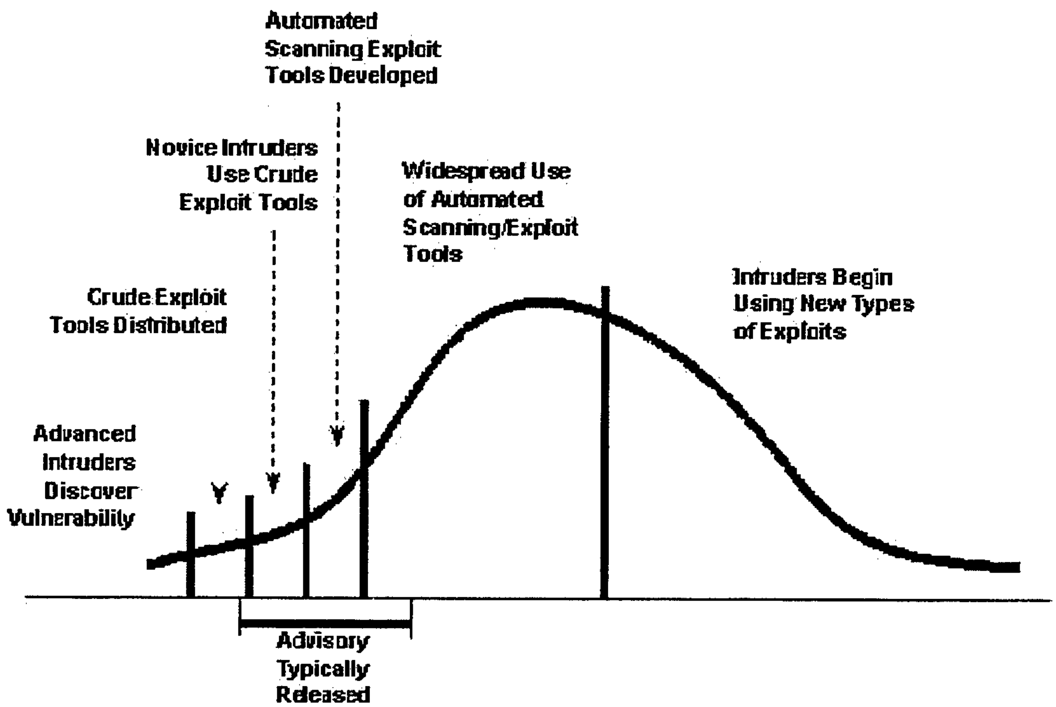

Figure 1. Vulnerability Exploitation Cycle (From Ref.CERT/SEI, 2000)

Figure 1 shows the Vulnerability Exploitation Lifecycle on the Internet. At the beginning, expert engineers or programmers discover a weakness through their experiences or jobs. Whatever their intent, they program crude code to either test for this weakness or exploit it. Soon either the coders themselves, or someone who has been able to put his hands on the code or new idea, begin to use it on systems throughout the Internet. The exploit is successful because the majority of sites are not yet aware of their weakness. It is at this stage that advisories from the various Computer Emergency Response Teams (CERT) begin to be published as the new attack spreads. The period between the advertisement and the patch by a vendor is highly variable, some patches are deployed within a few days while others are deployed in a month (Fithen, 2000).

At this point the tool is still technical: just having the code is not enough to use it properly; the user still needs to understand the inner workings of the target system and might have to "walk" the tool through to the end using other tools. Eventually, for either prestige or out of good intention, an expert will code it into a stand-alone tool or even a Graphical User Interface (GUI) -based application. The new code by the expert gives the

tool the push it needs into the mainstream "script kiddie" community and the tool's use becomes widespread. Now the new users do not need to know the underlying code, they can "stand on the shoulders" of the experts before them, benefiting from the expert's skill in coding, to attack a system that otherwise they wouldn't be able to compromise.

Eventually a patch or attack solution has wide enough distribution that the fixes stop the new attack. The successful use of the tool wanes sharply and attackers begin to look elsewhere for a new vulnerability.

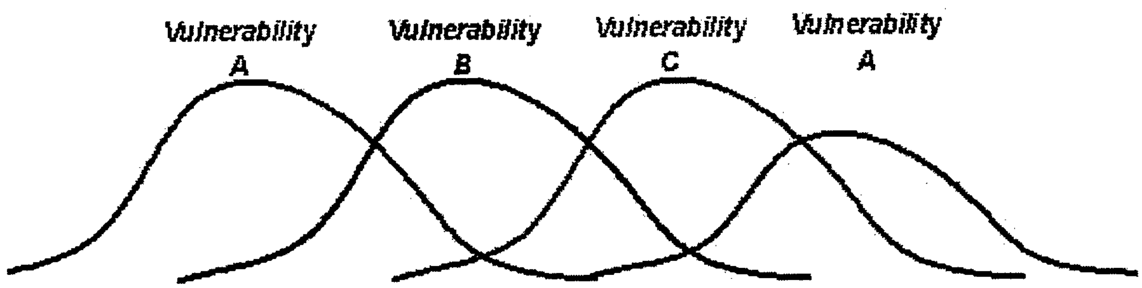

Figure 2. Vulnerability Lifecycle (From Ref.CERT/SEI, 2000)

Vulnerabilities are never completely eradicated on the Internet, and they are constantly being probed (McHugh, 2000). Figure 2 illustrates that in many instances the vulnerability has resurgence on the Internet after the initial decline. As the exploit becomes less notorious, and a new exploit is in the press, those responsible for system security might overlook the old vulnerability in future upgrades. All the solutions to old vulnerabilities have to be incorporated into security measures in the future. For example, Microsoft Corporation is up to Service pack 6 for Windows NT; it incorporates every vulnerability in service pack 3-5 inclusive.

One more factor in the renewed life of vulnerabilities is the massive rate of new users and systems. Vulnerabilities have an ever-expanding new range of computers to try their attacks on; the Internet is currently growing at 35% a year. "(McGuire, 2000). Compounding this problem is the practice of static IP address designation for the new DSL and ISDN customers. Static addresses for private citizens could be thought of as an easy target since security measures on private computers are hardly a huge concern.

Somewhere on the Internet it's almost a given that there are computers that are vulnerable to almost any tool, it's just a matter of finding them (Fithen, 2000). Private citizens do not spend the same amount of time on security than other types of groups because, "people don't consider computers as active instruments - rather like TV's and appliances" (Konda, 2000).

Statistically speaking, script kiddies and hackers have a huge chance of success of being able to find a weak system on the Internet. With well over 500 known vulnerabilities, sysadmins have no idea which ones to defend against and cannot keep up with the pace of the new vulnerability advances. "Sysadmins are overwhelmed by issues they face - it only takes one mistake. Time and tools are on the intruder's side to find the one weakness you have" (Carpenter, 2000).

B. APPLICATION/OS VULNERABILITIES

> Running system software straight from the box can be an invitation to hackers, as packaged software is often delivered with initial configuration settings [that] leave systems wide open to attack (Denning, 1999).

Applications have a wide range of vulnerabilities. Applications get more and more complex with every iteration, and the amount of code grows exponentially. Where once a few people could understand the totality of the code for a given application, modern applications have well over 20 million lines of code; groups are responsible for the security of only their specific aspect of an application. Modern code writing companies additionally don't feel any real pressure to make applications more secure (Longstaff, 2000). People demand that applications are easy, cheap, and quick over slower, more secure systems.

Computer companies develop their own code with known vulnerabilities and still place them out in the marketplace in "Beta" format. In the rush to be first to marketplace, they farm out their testing to the general public. If someone finds a flaw, he is under no real obligation to report it. A company under market pressure sometimes will fix as

many flaws as it feels necessary to guarantee only the proper running of the application, and end any improvement there. A common solution is to produce patches, which might fail to get the exposure they need.

Additionally, there is a great demand for backward compatibility in software. Backward compatibility forces some kernels to adopt vulnerable stances to handling security measures for processes and access to resources. Known vulnerabilities are still incorporated into new code to provide backward compatibility due to market demands.

Applications are usually shipped "out of the box" in the least secure mode possible. Many features of new products are vulnerabilities, such as Java and ActiveX, but rather than shipping products in the most restrictive mode, the onus for security is placed on the consumer. Sysadmins and civilian consumers have more emphasis to get the new product working rather than ensuring it is secure – and once it is working are loath to touch it.

Applications and Operating Systems contain vulnerabilities because security has never been the main concern for their development. Companies are more concerned to match market driven forces of speed and ease-of-use because people are not historically willing to pay extra for security. Vulnerabilities will always exist due to the extremely large amount of code in products and the impossibility of being able to test it completely. Coders discover and gather these vulnerabilities, incorporate them into tools, and eventually form them into self-executed programs that are easily used by any potential attacker.

C. INTERNET VULNERABILITY

The Internet, like software, is vulnerable because security was not an overriding factor in designing computer communications. Like applications, emphasis has always been on ease of use, data integrity and speed. At the beginning of the Internet Age, the integrity of sent data was the main goal; while in modern times, the consumers pushing the marketplace are far more interested in ease of use and low prices. Vulnerabilities on

the Internet are present due to the lack of attention to security and how the protocols are implemented.

1. Short History of the Internet

The Internet was originally called the ARPANET, which was started in 1969 by the Advanced Research Projects Agency (ARPA) of the Department of Defense (DOD). The ARPANET's main emphasis was the reliability and flexibility of the connections between computers, not security, and therefore very little thought was given to restriction of information flows. What little security that was present was mainly there to ensure connectivity (since every attempt to start a session was assumed to be legitimate). At that time, maintaining a reliable connection was very difficult between remote sites, and this much-needed reliability was one of the main emphases in the various protocol designs.

There were many reasons to trust other computers; computers were extremely expensive and maintenance intensive, and only highly endowed agencies were able to own them. Concurrently, access to a computer was almost proof enough of one's validity. In addition, the ARPANET consisted of universities and government sites where almost everyone knew everyone else: trust was not an issue in these small professional circles.

This open policy continued as the ARPANET grew and more features such as email, newsgroups and telnet services were added. Eventually the ARPANET proved to be more valuable as a communication tool than an end unto itself, as more people shared unrelated research (to ARPANET) over the net. By 1989 over 100,000 sites were connected to the ARPANET and the name was officially changed to the Internet.

By 1989, the small community flavor was gone. The first real virus, the Morris Worm exposed the impact of malicious code by spreading to roughly 10 percent of all the computers connected to the Internet in 1988 (Sullivan, 2000).

2. Packet Switching

The basis of computer communications assists in explaining why weaknesses are inherent on today's computer networks. In a perfect world, every computer would be point-to-point connected to every other computer with a dedicated wire and an Internet would not be needed. Of course this is impractical for two reasons: building dedicated lines between two devices thousands of miles apart would be extremely expensive, and every device does not necessarily need to be connected to every other device. (Stallings, 2000, p. 9) Therefore some type of switching needs to take place by all the other devices on the network where the resources are shared.

One of the earliest solutions was circuit switching. This was used for many years, and in some locations still is used in telephone networks. For a period of communication time allotted, two devices such as telephones are logically connected together over a shared line. Circuit switching has very rapid data transfer since there is no delay across the network. The downside is that the other devices logically separated from the network by the dedicated line of the two connected devices cannot communicate with any other devices.

The natural solution to circuit switching was to take advantage of the lulls when the two devices were not talking. Packet switching allows the line to be shared by multiple devices at once. The data is separated into small packets instead of a steady stream of data. These packets travel the nodes between the two devices, and not necessarily over the same nodes either. The connection can be used much more efficiently than circuit switching since communication can take place constantly.

This is one basis for the weakness in computer communication. Since communication is not constant over dedicated lines, a series of protocols has to be used to encode messages. The protocols encode and decode these messages, sending them off using different protocols that have inherent vulnerabilities.

3. TCP/IP

> The transition to TCP/IP was perhaps the most important event that would take place in the development of the Internet for years to come. After TCP/IP was installed, the network could branch anywhere; the protocols made the transmission of data from one network to another a trivial task (Hafner, 1998).

TCP/IP is the protocol for the World Wide Web and stands for Transmission Control Protocol/ Internet Protocol, and is actually two different protocols working in unison. IP is the infrastructure that separates each individual node with a unique address, similar to a postal address. TCP works on top of this infrastructure, separating the data into "packets" that can be routed to their final destinations. When TCP/IP was adopted, along with the Domain Name System, "every address would include levels of information representing, in progression, a smaller, more specific part of the network address." (Hafner, 1998)

TCP/IP has been adopted because it was one of the original protocols that incorporated a method of error checking and is very reliable over multiple systems and architectures. It has some very distinct characteristics, outlined in several RFCs about how one starts, continues, and completes a connection with another computer. RFC971 explains IP, the basis for all communications on the Internet.

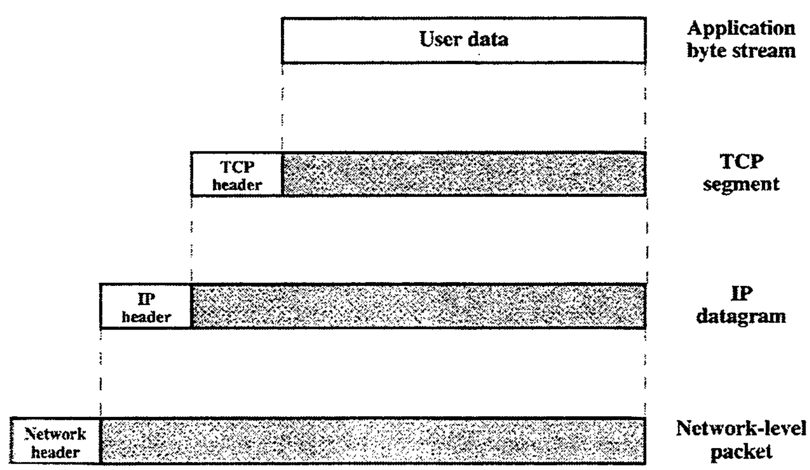

Figure 3. Protocol Headers (From Ref. Stallings, 2000)

Figure 3 gives a simple example of how data is prepared to be routed using TCP/IP and shows the different headers that are placed upon base data. This is how the Internet works: starting with the application data level, TCP, IP and the Network level protocols all add extra "routing" instructions to ensure the data reached its proper destination. Within each header are specific instructions for each layer, like envelopes within envelopes. When the packet reaches the other end, each layer strips off its respective header, using the stored information to pass off to the next layer. The packet is "self-sufficient" in that all instructions are incorporated into the packet and it is sent off by itself. Once it goes outside its security zone, the packet is vulnerable to being altered, captured, or spoofed. Of course, data fields in the headers can be altered on purpose at conception as well.

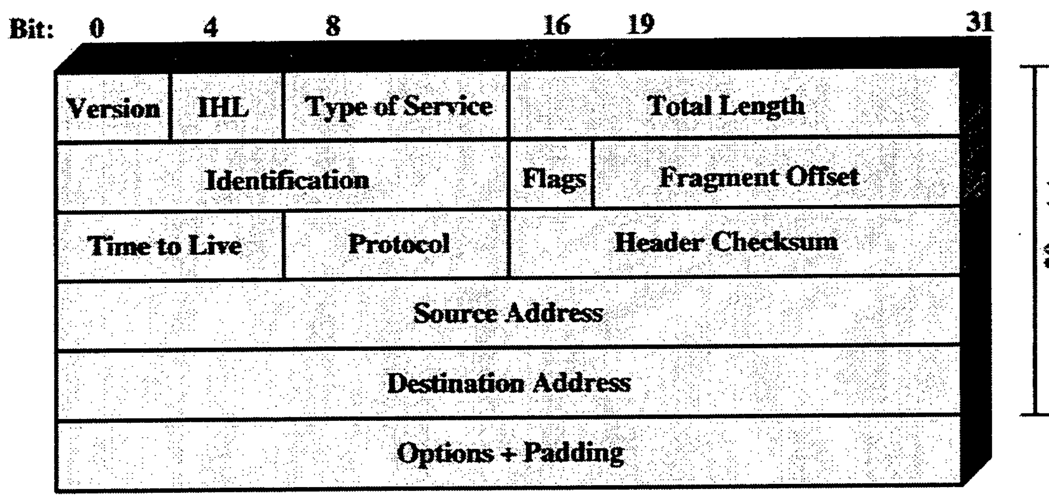

Figure 4. IP Header (From Ref. Bernstein, 2000)

Figure 4 shows an IP header, which provides information to routers to help send the data to its destination. The most notable fields in the header are the source and destination address fields. Many IPv4 routers accept IP headers implicitly and do not check for validity, and anyone wishing to change the source address to any 32-bit number is welcome. This weakness is the starting point of many attackers wishing to hide their

identity. One could also change the other fields, such as Fragment Offset, to any value one wishes, which causes another error from malformed packets.

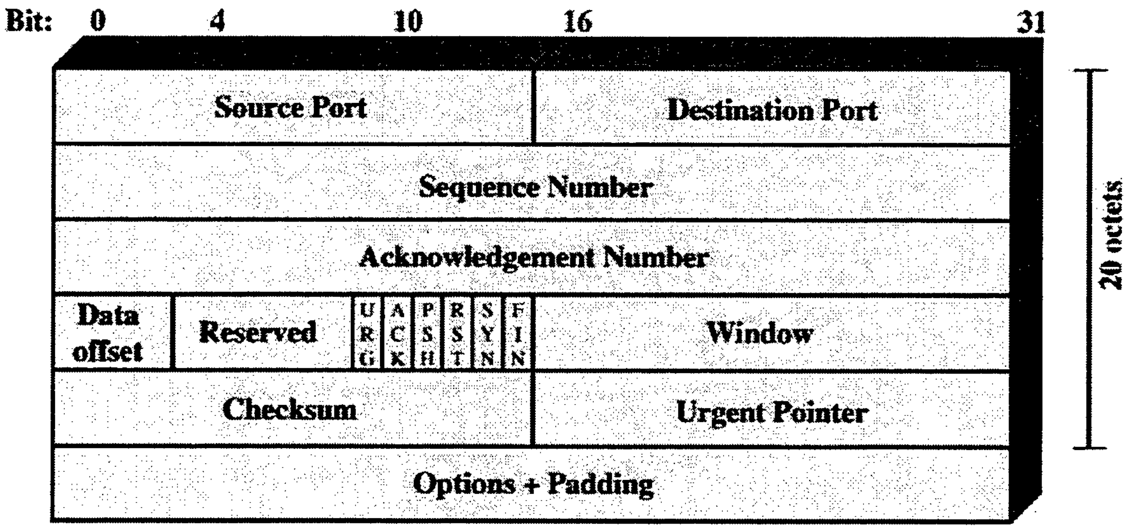

Figure 5. TCP Header (From Ref. Bernstein, 2000)

Figure 5 is the template TCP header that is placed in front of the IP header. Similar to the IP header, TCP adds more specificity to the packet in the form of outlining the communication ports used, sequence number of the communication stream, and several signal flags. Like the IP header, the TCP header can be manipulated to cause errors or induce the target computer to act in certain ways.

4. Examples of Common Exploits in TCP/IP (and why they work)

Two of the most common TCP/IP exploits give a good example of how altering a few bits in a header can have malicious consequences. They also reveal how inherent trust is taken advantage of and how protocols can be used against themselves.

a. Initial Handshake

Many administrative and malicious tools on the Internet take advantage of the handshake sequence of TCP/IP. When two computers try to talk over the Internet using TCP/IP, a "three way handshake" is initiated. Each phase of this handshake is vulnerable to information gathering, spoofing, or denial of service tools.

To open a dialogue between computers, several packets of data need to be sent in a certain order between them. The source computer sends an initial packet requesting a dialogue, which is signified by a SYN flag in the TCP header. The target computer sends an ACK and SYN flag in return to signal that it is ready to receive data and to go ahead and do it. The source computer then sends ACK and starts sending data.

If a computer sends a SYN signal and loses its connection, the target computer will still send its reply and wait for a response. This is a weakness where a computer could "time out" a target computer by simply sending a constant stream of SYNs. The target computer will wait for the final handshake for each SYN packet received until its buffer is filled, thereby blocking any legitimate traffic to the target computer. This is known as a SYN flood attack, a type of Denial of Service (DoS).

In another exploit, a computer can send connection attempts to every port on a target computer and log any replies, effectively scanning and enumerating the computer. Even a negative response can help determine the characteristics of a computer.

Taking advantage of the initial handshake uses a trusted relationship in a malicious way. The target computer is simply trying its best to connect to what it perceives as an honest attempt to communicate.

b. Spoofing

As mentioned earlier, one of the easiest weaknesses of TCP/IP to exploit is to change your identity. This weakness is taken advantage of by many exploits on the Internet and is the first step in hiding your identity to an Intrusion Detection System (IDS) or the authorities. Instead of changing your identity to a random IP address, one can assume the identity of one side of a legitimate communication.

Spoofing also involves trying to hijack a session from another legitimate connection. TCP uses the Sequence Number to keep track of an open communication session. If one can guess the next semi-randomized number of the sequence, one could jump in the middle of a legitimate conversation and hijack it.

The main vulnerability in TCP/IP is the implicit trust in the system. There is little validation of headers, which opens the door to several types of vulnerabilities and their exploits. To put it simply, "host names, IP addresses, and hardware addresses are not sufficient to create any basis for trust on the Internet"(Atkins, 1997), due to the ease of being able to modify them.

D. NETWORK ORGANIZATION

The structure of networks also has vulnerabilities that can be taken advantage of. The Internet is divided into domains and each domain is given a range of IP addresses that it can use. Each domain also has an IP address that is owned by the router. The router address is used to send and receive packets for the domain that it represents.

Routers use common commands for maintenance purposes. For example, it is necessary from time to time for the router to talk to every node that it is responsible for, to keep track of every computer. Instead of methodically sending packets to every single computer or node on the domain, the router uses the broadcast IP address – sending the same message to every computer at once. Usually this is a ping, which requests a signal

back. The "broadcast" ping, originally implemented for network control, has been converted into an intelligence gathering device and an attack tool by hackers. Many early developed commands still widely used for LAN maintenance have been subverted by malicious code in ways the original designers never imagined.

1. Coordinated/Distributed Attacks

> ...there is essentially nothing a site can do with currently available technology to prevent becoming a victim of, for example, a coordinated network flood (CERT, 1999).

A coordinated attack is an attack that uses several computers in conjunction to exploit a vulnerability. Coordinated attacks also are used to magnify certain attacks, such as a broadcast attack. Many coordinated attacks and distributed attacks take advantage of vulnerabilities in both applications and the Internet together.

One of the simplest coordinated attacks is a distributed denial of service. Instead of using a single computer to flood a target, multiple computers are used in conjunction to greatly increase the amount of malicious packets. In addition, if the computers spoofed their IP source addresses, the target computer would not know who the true attacker was. This example uses the network vulnerability of the broadcast ping and the IP protocol vulnerability together to form an indefensible attack. In this example, the target domain has no way to stop the exploit, since every source address is false. The solution is for the source Internet Service Provider (ISP) stop fake packets from leaving its site.

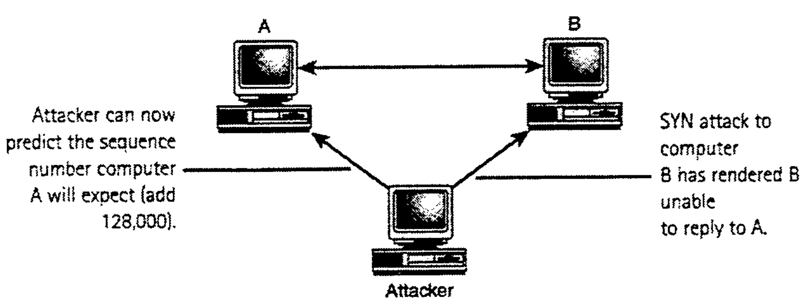

Figure 6. A Coordinated Attack (From Ref. Northcutt, 1999)

Coordinated attacks can also be used to attack a system from two different directions, taking advantage of the vulnerabilities in the communication protocol between two computers. Coordinated attacks can be used to manipulate trusted relationships between computers. As shown in Figure 6, one could both spoof one computer and flood another at the same time to spoof a connection. Kevin Mitnick, easily the most famous hacker of the 90's, used this coordinated attack to gain his notoriety.

E. CONCLUSION

> System administrators out there... are aware that security holes exist in their systems, but they see the alerts coming out daily and are overwhelmed by sheer numbers (Lemos, 2000).

Internet and application vulnerabilities are always existent. New vulnerabilities are always being discovered and coded, and unfortunately, "threats are winning faster than defense" (Shimeall, 2000). Vulnerabilities are present due to the assumed "trust everyone" relationship between computers, the complexity of modern code, and the way the Internet was originally designed.

Additionally, new systems and applications are put in place with safety defaults turned off. They are designed with speed and ease of use rather than security in mind. Patches to known vulnerabilities are not put in place soon enough, but even with timely patches, "sites are almost always taken over by known attack scripts that should have been secured" (McHugh, 2000) due to other sysadmin's priorities.

Fortunately, several high profile attacks in 1999 and 2000 have raised the level of interest in security. The public and government are gaining more interest in the subject with headlines such as "Its Time for Uncle Sam to Foil a Cyber Pearl Harbor" (Pinkerton, 2000) and the myriad of articles surrounding the DoS attacks against the e-vendors and the "I Love You" Worm. The government has also developed the "National Plan for Information Systems Protection" which is a partnership plan between the government and business to confront security issues more seriously.

Many feel that exposure of vulnerabilities will make the Internet and applications stronger, "just as our immune system is strengthened by exposure to disease"(Markoff, 2000). Weak systems will be exposed and taken down while the strong will adapt. Every computer system has vulnerabilities. True security might depend on making yourself that much tougher than your neighbor and utilize the "Low-Hanging Fruit Paradigm" (Northcutt, 1999).

Finally, despite all the code and vulnerabilities, social engineering still remains one of the most vulnerable parts of a system. An insider, weak passwords, and not enforcing physical/logical security measures all negate any security posture.

In conclusion, the Internet community is dealing with the vulnerabilities inherent in all of the protocol levels. Malicious coders take advantage of the "assumed trust" relationship between computers. New vulnerabilities will always have a short advantage over legacy systems. But now with the new interest in security and the new measures that are slowly being developed, such as IPv6, security will inherit the same importance as simplicity and speed one day. However, constant vigilance is still needed at all levels to decrease the presence and effect of vulnerabilities as the foundations of the Internet are slow to change.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODS OF THE ATTACKER

"We often give our enemies the means for our own destruction" - Aesop.

This chapter explains how an attacker might use open source tools to select and attack a given target. Three popular methods are described: (1) Exploit, (2) Trojan Horse, and (3) Denial of Service. While there is no one overriding way an attacker can attack, by investigating all three of these methods, one may infer much about most of the other forms of computer attack. This is not a comprehensive guide of the best tools on the Internet, but rather a sample of what an attacker can download today off the Internet and what they can do with them. The chapter concludes with methods of finding open source tools and instruction.

This chapter is organized so that the reader becomes familiar with three types of attack. First an overview of the attack is presented, followed by a more detailed view of a specific tool that performs tasks needed in the attack. The tools have been picked based on their popularity in the major hacker/cracker sites and upon the personal experience of the author. The difficulty in providing examples of open source code is the specificity of each tool: most tools target only specific operating systems, mainly in the case of "end-game" tools: the buffer overflows and application/OS exploits.

The goal of this chapter is to demonstrate how open source tools can be used in an integrated way to gather information and complete an attack. Most tools imply a non-trivial amount of knowledge to use (Nelson, 1999). Simply having access to the tools does not equal the ability to use them. Knowing how to use the information from the tools is a necessary skill. Just downloading these "example" tools and blindly using them

doesn't guarantee anything; however, in the right hands, a system can be used against itself to reveal information that otherwise would remain hidden, or to find vulnerabilities that the manufacturer either accidentally overlooked or implemented into the code.

The chapter concludes with a section on how to find open source tools. Rather than give an exhausting, easily outdated list of the most popular tools, the section discusses where the open source tools reside on the Internet.

A. NECESSARY STEPS FOR AN EXPLOIT ATTACK

An exploit attack is defined as an attack to gain root status or the ability to execute code at root level on a remote system (McClure, 1999). According to several sources both in print and online, there are roughly eight steps needed to complete an exploit attack (Maximum Security, 1998). The steps are: (1) Footprinting, (2) Scanning, (3) Enumeration, (4) Gaining Access, (5) Escalating Privilege, (6) Pilfering, (7) Covering Tracks, and (8) Back Doors (McClure, 1999). The first four steps involve three areas: finding, targeting, and attacking an objective. The scope of this paper is restricted to the first four steps, as after gaining access, each step is highly specific to the system being attacked.

1. Footprinting

> [Sam Spade] is a multitalented network query tool, with some extra utilities built in to handle spam mail. It provides the typical utilities such as ping, traceroute, whois, finger, etc. (PCWorld, 2000).

The first step of any attack is determining a target of interest. This could be any system that has a particular known weakness that is searchable, or a specific target. Footprinting is the easiest of all the steps; it is basic intelligence gathering on a target generally using legal and accepted search tools (McClure, 1999). There are several open source tools that help determine basic open information on a target.

Footprinting can start with a query as simple as a Yahoo, Hotbot, or Google open search entry and these usually will bring up the homepage of the target. For example, entering "Naval Postgraduate School" will return the *www.nps.navy.mil* homepage on any search engine. There are also several hacker search engines such as *Dogpile.com*, which allows one to check several webcrawlers at once. All of these engines give the attacker the starting location of the target, the first step in isolating the zone of interest.

From the homepage of the target, one can use any browser such as Internet Explorer (IE), to find an IP address of your target by resolving the Domain Name System (DNS) name to an IP address. For example: the DNS entry for *www.nps.navy.mil* resolves to 131.120.251.13. By using another public site, *ipcheck.dragonstar.net*, a service that resolves IP addresses, reveals that the Naval Postgraduate School has responsibility for a whole Class B network: all the numbers from 131.120.0.0 to 131.120.255.255, roughly 65,000 IP addresses. An attacker now knows what range of IP addresses he needs to focus on.

Another useful public footprinting tool is the "whois" feature – every registrar (such as Network Solutions Corp.) is responsible for listing system administrator (sysadmin) contact information for public access (*www.crsnic.net/whois*). The sysadmin information usually includes the address and phone number of the domain site (where the

computers are located) and the sysadmin name and number. In addition, any extra domain names are listed, which can reduce the number of addresses that need to be checked by an attacker. Attackers use the information to get an idea of a geographical location and to get a range of what IP numbers they have to investigate. (Bernstein, 2000) This is a common social engineering break-off point, where hackers would use the name of the sysadmin to try and trick a common worker to reveal a password or userid, or even set up an account (Littman, 1996).

Many open source footprinting programs include multiple tools and features. One of the more popular open source tools is called "Sam Spade", which can be found at *www.samspade.org*. The advantage of Sam Spade is its Graphic User Interface (GUI) that can be set up to systematically get information using a whole range of tools at once. Included in the software is an extensive help file and online help page, to explain any results one might have questions on, and instruct different methods of footprinting (Atkins, 1999).

Footprinting is the starting point for any attack. With footprinting tools, an attacker finds the domain of the target, the range of IP addresses in the domain, the sysadmin POC with name and number, and the routes the line of communication goes through to reach the target. Like a burglar "casing" a house, footprinting determines where the house is, who is responsible for its security, and the location and number of windows/doors into the house (McClure, 1999).

2. Scanning

> ...[T]he 'to current resident' brute force style of bulk mail is an almost perfect parallel [of scanning]...just stick a message in every mailbox and wait for the responses to trickle back. We send a blizzard of packets for various protocols, and we deduce which services are listening from the responses we receive (or don't receive) – Fyodor (2000).

Scanning tools are used when you have determined the rough outline of your target from footprinting. The scanning step is used to determine which computers are "alive" and which services the computer on the domain is using. These tools are used to get an exact center: determining the types of operating systems and the various ports that are open to attack (Bernstein, 2000). Open source scanning tools use both frontal loud attacks and subtle quiet attacks to get information.

TCP and UDP services can be scanned on a remote computer by sending false and legitimate packets. By taking advantage of the three-way handshake for each TCP port, an attacker can determine which services are running on the target. There are many methods of scanning a computer for different services, and each can discover a small bit about the target computer, even whether it answers the scanning packets or not (Fyodor, 2000). With a summary of open TCP/UDP ports and how the computer reacts, an attacker can determine the operating system and applications. By knowing the details of the system an attacker is up against, he can search the Internet for the known vulnerabilities for that particular system or applications.

Scanning can be done by brute force: every computer has 65,535 ports, and each can be assigned an application. The majority of ports under 1024 have already been assigned; for example, "port 80" is the http port, the Internet port. Every computer that

surfs the web has “port 80” open. “Nmap” can systematically determine all the ports open on a target and use the results to determine the services running on the computer.

The most powerful and popular scanning tool developed is “nmap”, produced by a person who calls himself Fyodor (Bernstein, 2000). Nmap can be used on Linux and Microsoft systems to systematically scan a range of IP addresses and ports. Nmap has been described as “one of the most powerful information-gathering tools available ...to both attacker and defender” (Northcutt, 1999). Nmap can be scripted to scan in different techniques to fool Intrusion Detection Systems (IDS) (Fyodor, 2000). Using these two features of nmap together, an attacker can discover the target’s router or operating system while hiding the scanning process.

In conclusion, if footprinting is “casing the joint”, then scanning would be the rattling of all the doors and windows after driving to the target. Scanning a target tests every port to determine which services are running on the target. Attackers use scanning as a stepping-stone, to focus attempts on finding services that are running on the target.

3. Enumeration

> [Nmap] provides all the information needed for a well-informed, full-fledged, precisely targeted assault on a network. Such an attack would have a high probability of success and would likely go unnoticed (Northcutt, 1999).

Enumeration determines the group and user structure of the target domain or spotlights a weakness in the security layout of the target system. The key to enumeration is to find ways to get access to a root level process, which allows an attacker to run or set up any code that they wish. This step discovers structural weaknesses in the defensive

matrix of the target domain by determining the specific applications or hardware being used by the target. The point is to gather enough information about the computer so attacks can be specific to the current services.

Perhaps the easiest way to obtain a simple enumeration on a target system is to use the *www.netcraft.com* public site. Netcraft will enumerate any site as best as it can – and give the user the target's browser application and system platform version if possible. Nmap can also be used to enumerate a target system. Based on responses and target ports open, nmap can give its "best bet" of what the attacker is up against. For example, "port 139" (netbios) and "port 80" (http) together usually indicate a Windows machine.

One of Nmap's main value as a tool is its internal database of responses that provides a script kiddie what type of system is on the other side. Nmap parses and computes the target by the algorithms that were coded by Fyodor, the programmer of nmap. The answer can be used to give an attacker a starting place to look for known open source vulnerabilities for that particular platform and the applications it runs.

An attacker enumerates a target to find more detailed information on a system. Using enumeration tools provides an attacker with several starting-off points to attempt gaining access. Knowing the version and type of applications, allows one to search for publicly known weaknesses on those applications.

4. Gain Access

Once the target has been enumerated, an exploit is used on the vulnerabilities that are associated with the applications and/or operating system. Finding the exploits involves searching the Internet for tools specifically related to the target. There are a plethora of techniques and tools available on the net to gain entry, including buffer overflows, operating system errors, application flaws, etc. The meaning of "success" of these exploits could range from being able to read a locally held file to owning a root level account on a domain server, with free reign on all the organization's resources. The tools and techniques used to gain access get very specific for each platform and purpose, as many vulnerabilities are usually effective only to specific applications.

For purposes of demonstration, a buffer overflow will be described. Buffer overflows are some of the most common exploits on the Internet (Network ICE Corporation, 2000). A buffer overflow is "what happens when you try to stuff more data into a buffer than it can handle" (*www.jargon.net*). In other words, it attacks a lack of bounds, checking on the size of input being stored in a buffer array. In simple terms, it is a response that fills a memory location beyond what it can handle. For example, if a computer asks for the year on a form, where a short integer and a response of four digits is expected, the attacker will instead provide an overwhelming response of 150 numbers. Buffer overflows only work in code that does not check the boundaries of its arrays.

By writing past the end of the allocated space, the attacker can make changes to the memory stored next to the array on the stack. If the code is written poorly without error correction, the 150 numbers will swamp the executable code into allowing the user

to point a pointer into alternate memory location. As shown in Figure 1, in the other memory location the attacker will have special code that he wants to implement. Most of the time the hacker will introduce code to start a "shell", which allows the attacker to run processes as root. Buffer overflows are not easy to find, and are usually the result of intensive reverse engineering to find the exact offset to introduce onto the stack (Cowan, 2000). The open source tools on the Internet are usually restricted to pointing out where the overflows can be introduced, and leave the type of attack open to the attacker.

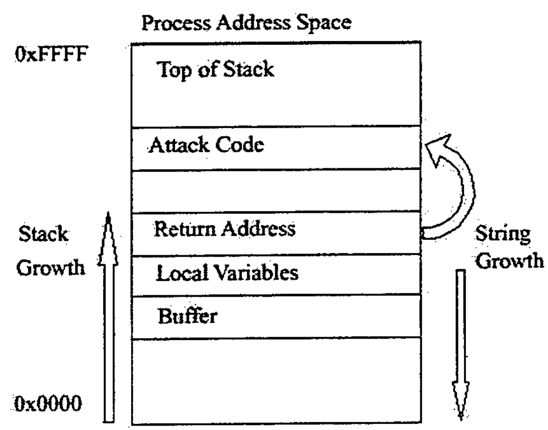

Figure 7. Buffer Overflow (From Ref. Cowan, 2000)

One recent buffer overflow, introduced on 18 JULY 2000, specifically targets MS Outlook and Outlook Express Email Clients, two very popular applications that Microsoft has incorporated into their "Office suite" bundle. If one has enumerated that the target system uses MS Outlook, this exploit might be able to be used. (This exploit can be found at *packetstorm.security.com/0007-exploits/outlook.advisory.txt.*) This buffer overflow exploit advisory outlines the method in "smashing" the stack by entering "an exceptionally long string directly proceeding the GMT specification in the Date header

field." The buffer overflow occurs when the target user previews or views any email in MS Outlook. The exploit also gives an example of using this overflow, demonstrating how one can force the target computer to download and execute a file from the web. One can easily see how a malicious attacker can do anything within the target system that has this vulnerability (CERT, 2000).

With any open source exploit, the attacker is hoping that the security manager has missed the patch for the vulnerability or has been too busy to upgrade the software. If the security manager has been on the ball, the attacker will simply search for a new open source tool that can affect the target system in another way, using the information gathered in the enumeration step.

5. Conclusion to Exploit Attack

Exploit attacks can take many different forms, buffer overflows only being one. Just having the tool does not guarantee success: knowledge of how to implement the information given after each step makes the attacker dangerous. Many of the steps can be combined, as shown with nmap in the scanning and enumeration phase.

Once the attacker has gained entry, there are many steps he/she needs to make which involve setting up a user account, hiding their tracks, and making a back door so they don't have to hack in again. Each of these is highly specific for each individual operating system and not a focus of this thesis. There are many open source tools and advice on the Internet where one can learn and be taught to discover how to do those things.

There are also several tools on the Internet that combine many steps at once, such as "SATAN" and "ISS". Both scan and find weaknesses in systems automatically without an attacker needing to do extensive research. Many attackers will not use these "multipurpose" tools because of the large signature they leave on the targeted system. Both SATAN and ISS are very popular and all the new IDSs can quickly identify when they are being scanned by them (Maximum Security, 2000). To remain hidden, attackers use the slow stealthy methods that keep them "below the radar screen" of the targeted system's security measures. Sometimes the easiest tools to use have the most popular signatures because of their notoriety. Knowing the intricacies of the computer languages and commands outweighs the advantage of access to the newest tools.

B. TROJAN HORSE ATTACK

Trojan Horse attacks can be viewed as a shortcut to an exploit attack. Rather than breaking into a system, an attacker can trick a target user into running a malicious program. Trojan Horse attacks refer to the battle tactic of the Greeks at the end of the siege of Troy outlined in The War At Troy, written by Quintus. The modern Trojan Horses are based on the same premise – the recipient is getting more than they bargained for. Trojan Horses are extra code, piggy-backing on legitimate code to gain entry into another system. No drawn out scanning and enumeration are required since the person fooled runs the malicious code under his access level on the system. Trojan Horses are a method of gaining access – only the type of code that the attacker chooses to run limits the range of damage.

For this chapter, Trojan Horses refer to the practice of installing a stand-alone Remote Administration Tool (RAT) program. A RAT is an extra "hidden" process on a target computer that allows an outsider access. Every time the target computer is started, the hidden RAT is started as well, opening a port on the computer to allow an attacker to send any command to the target computer. To infect a system, an attacker has to get the target user to run the seemingly innocent Trojan Horse application, and then the RAT program will automatically burrow itself into the operating system.

The RAT Trojan Horses are composed of two parts: the server and the client. Since the attacker will be communicating with the server, it is placed on the target. The attacker uses the client to talk to the server(s). The attacker attempts to load the server on as many systems as possible. Communication between the client and server is by TCP/IP on an attacker-defined port number.

Trojan Horses can also be used to ensure access to a compromised system. Some attackers set up Trojan Horses and then hide them in the system and wait a month or longer before using the attack program to ensure that the Trojan Horse is incorporated into the monthly system-wide backup. In case of being discovered while using the system later on, their Trojan Horse will be reinstalled with the back up (Dittrich, 2000).

1. Trojan Horse Walk Through

> ...[Sub7 is] the most popular [and]... is the most dangerous Trojan, with several powerful "hacker" capabilities. (Network ICE Corporation, 2000)

Every Trojan Horse attack comes down to getting the code onto the target. By far the easiest method is to attach the malicious Trojan Horse to an innocent looking email. One of the oldest Trojan Horse spreading methods was animated holiday programs that would install the malicious code while the user played a simple holiday game or watched a holiday animation (Bernstein, 2000). For the majority, an unknown executable file from anyone not completely trusted would be deleted without running. However, even with all the press warnings, it would still be a very effective method, proving once again that the weakest link is always the user[1](Finley, 2000).

There are three popular Trojan Horse RAT programs on the Internet: Back Orifice, NetBus, and Sub7. To describe a Trojan Horse server/client program, Sub7 and several "masking" programs are discussed. Because the main concern of an attacker who is restricting himself to an email attack would be hiding the true nature of an attachment, several tools must be used to change the nature of the malicious file.

The Sub7 file can be found at its homepage (*http://subseven.slak.org*). It is a relatively small application – 1.35MB that includes the client, the server and an additional server editor. The installation file that goes onto the target, the server, is 373KB – easily hidden in a much larger video file. Included at the Sub7 site are explicit instructions on how to use the product and several suggestions on how to successfully infect a target. One of the methods suggested is to use separate programs to hide the true

[1] The person who made the "I Love You Too" worm, named after the more famous "I Love You" worm, supposedly invented it just to showcase that even though there was wide spread warnings not to open suspicious email attachments, people still did in droves, even after being infected with the "I Love You" worm.

nature of the server. The first is called "joiner" that an attacker can use to attach the server to another file.

A programmer who calls himself "Blade" developed joiner. The site (*www.come.to/soul4blade*) has several Trojan programs as well which can be used in conjunction with Sub7. Now that many users have exposure to Trojan Horses, Blade has developed a tool called "tHing" that is only 8k large, and it basically creates a very small opening to allow the bigger Sub7 Trojan to be installed at a later date. The joiner program attaches the server to any file an attacker chooses, including sound, photo, and video files. After using joiner, the file is changed to an executable and a new icon is placed on the attachment.

To change the icon, an attacker can use the Microangelo98 program from (www.impactsoft.com) – a free icon-editing tool to change the appearance of a file. Finally, the attacker sends the trojaned video to a target as an attachment hoping it is opened out of curiosity. If sent to enough people, the attacker is almost certain to have one or two open the video.

Upon opening, the server attaches itself and "hides" in the system registry, the heart of the operating system. The server can be also configured to reinstall every time the computer reboots making it difficult to get rid of, and it can also change its name to mimic a vital program in the operating system. Very few untrained users would erase a file named "Winkrnl.dll" – or even know that this could pass for "Windows Kernel Dynamic Linked Library" – a bogus file (Bernstein, 2000).

The attacker uses the client to scan the targets to determine if the server has been loaded. The server and client communicate on a high numbered port, well into the

1000's, thus evading the well used ports and services. Usually the first step of a successful Trojan Horse RAT attack is to close off access to anybody else trolling the Internet for the same open ports. As discussed in *soul4blade.com*, there are several tools that just scan for certain open ports that designate a Back Orifice or Sub7 RAT servers.

C. DENIAL OF SERVICE ATTACK

> Distributed Denial of Service Attacks have recently emerged as one of the most newsworthy, if not greatest, weakness of the Internet (Todd, 2000).

A Denial of Service (DoS) is an attack that stops the function of a computer or a system from communicating with the outside world (Dittrich/DoS, 1999). DoS attacks usually take one of two forms – resource starvation or resource overload (RFP, 1999). In its base form, a denial of service could be considered the same as unplugging the computer from the wall: DoS keeps the target from being able to continue with legitimate work. "A denial of service attack is characterized by an explicit attempt by attackers that restrict legitimate users of a service from using that service" (CERT/DoS, 2000).

Usually executing a DoS attack is simpler than an exploit attack. In essence, all one needs to do is footprint and scan a target. Once a target is found and determined to be susceptible to a denial of service, an attack can be made. Denial of Service attacks take many forms – for example, www.technotronic.com/denial.html lists over 25 different methods of instigating DoS attacks. The most common form of DoS is packet flooding, briefly described in Chapter II. If one continually sends SYN packets to a target, using a

simple packet crafting application, the target's buffer will fill up and block any legitimate traffic while waiting for more packets.

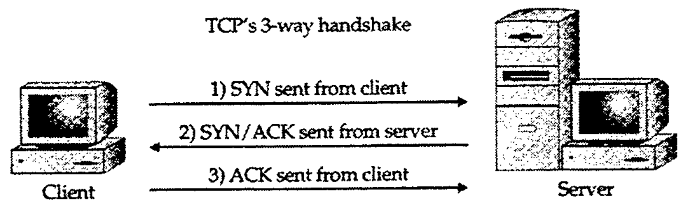

Figure 8. TCP/IP Three Way Handshake (From Ref. Northcutt, 1999)

Sending a massive amount of legitimate traffic could flood a target. An attacker can send packets to a broadcast network and have all the replies sent to the target, thus multiplying the target's traffic by a factor of 50 to 200. Craig Huegen of Cisco systems demonstrated this attack, called a smurf attack, using half a T1 connection. T1 connections usually run at 1.5Mbps and Mr. Huegen used 768Kbps, roughly the equivalent an attacker would be able to use in a cracked system. By bouncing the 768Kbps worth of packets to two broadcast networks, he was able to produce 67.5 Mbps of traffic to a third target, multiplying his traffic by a factor of 88 (Huegen, 1998).

1. Example of Denial of Service

One of the more nasty open source tools on the Internet is "TFN2K", a Distributed Denial of Service (DDoS) Attack tool. TFN2K stands for Tribe Flood Network 2000, the improvement to the original TFN attack tool written by a programmer called "mixter". TFN2K, as opposed to a simple attack, uses multiple computers to carry

out a DoS. The attacker sets up on a network of computers that can multiply the amount of packets that can be generated than by a single computer. Figure 3 shows a distributed denial of service attack in action.

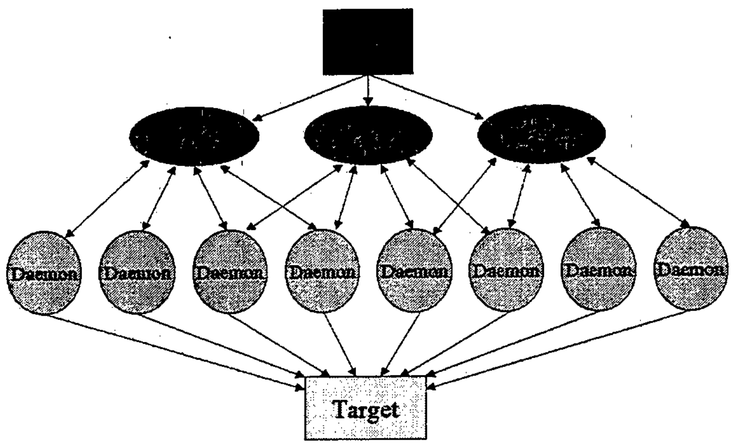

Figure 9. Distributed Denial of Service Architecture (From Ref. Bernstein, 2000)

TFN2K, at its core, is composed of two parts, the masters and the daemons. The master sends commands to the daemons that reside on different computers. The communication is one way from the master to the daemons, which helps with the anonymity of the master. Previously the attacker had to break into multiple machines and load the daemons on the target computers, by either Trojan Horse attacks or Exploit attacks. When sufficient amount of computers have been loaded with the daemons, the attacker crafts an attack strategy on his master computers.

With TFN2K, the attacker can pick between a myriad of attack techniques and strategies. The attacker can change the source IP numbers to either hide the master's identity and/or implicate another site. To the target under attack, it would appear that a

huge range of random sites were demanding resources, sending malformed packets, and pinging all at the same time.

Many sources feel that the attacks on Ebay and Yahoo earlier in 2000 were from a DDoS attack. In one attack on the University of Minnesota, an attacker loaded daemons into over 2,200 computers and was able to take down the system for three days (Dittrich, 2000).

2. Conclusion to Denial of Service Attacks

Simple denial of service attacks can be made from almost any computer, without needing much skill. Most of these attacks could be tracked to its source or stopped at the target's ISP through implemented security measures and patches. The reason why Denial of Service is not more common is the high degree of chance there is of being caught by the authorities if the attacker does not cover his tracks well. Few people have the skill to hide their identity on the Internet effectively (Northcutt, 1999).

Distributed Denial of Service Attacks involves both Trojan Horse Attacks and Exploit Attacks to set up an attack network. Skill is needed to hide the daemons and hide the fact that target systems have been infiltrated.

Theoretically, there is no defense to a DoS; an overwhelming amount of legitimate traffic can induce a denial of service. Several ISPs now block certain features that could hinder certain DoS attacks, but the dedicated attacker simply would then attack the target's ISP. Because of the lack of skill it involves, Denial of Service attacks are usually looked upon in disdain by the hacker community (see www.packetfactory.com and

lectures at DEFCON 2000). It is a brute force attack that has very little finesse. However, it still poses a major threat to sites because of the lack of ability to counter or block it. (Dittrich/DoS, 1999)

D. SOURCES OF ATTACK AND TOOL SOFTWARE

> There is an increase in attacks after an advisory [from CERT] – the advisory tells the intruder community when a new exploit is out there (Carpenter, 2000)

Rather than give an exhaustive list of tools that would more than likely be outdated by the publication of this paper, this section discusses the methods of how script kiddies find modern tools and techniques. Attackers and script kiddies rely upon the expert programmers and engineers to find the vulnerabilities and then develop tools to exploit the vulnerabilities. The security industry and security focused websites give valuable starting points to the script kiddie attacker. A prospective script kiddie attacker can "ride the wave" of the vulnerability exploit lifecycle outlined in Chapter II. " Script kiddies are happy with any way in [and] you can almost always get in" (Fithen, 2000). By attempting a new exploit quickly, an attacker might be able to sneak in before the sysadmin fixes the weakness. There are several government and industrial sponsored sites that list all the new notifications on a daily basis. There are also openly hacker-friendly sites that list all the new tools and exploits on the Internet as well. It's only a matter of research and motivation: checking sites every day to see what is new on the Internet.

One of the best sources of vulnerabilities and exploits is *www.securityfocus.com*, where vulnerabilities are outlined by individual platforms. The site also has a search engine that an attacker can use once the target has been enumerated. It has daily updates on the computer security field and links to a wide range of tool providing sites. It's far from being the sole provider of security related information, but to the author it appears to be the user-friendliest. Lists of popular starting websites, Usenet newsgroups, and newsletters that an attacker could use to start information gathering are given in the "Links" pages of almost any security related website.

An attacker can also attend the growing number of annual hacker conventions around the country to meet experts and learn how people both attack and defend. Experts from both sides of the security fence, both "black hats" and "white hats", give demonstrations and lectures. Included in the DEFCON 2000 convention was a room specially configured for people to practice hacking in a "competition" environment, with a high learning curve. Although usually hosted more for its social aspects of the computer community than anything else, conventions give opportunities for prospective attackers to learn more about their craft.

Additionally, IRC (Internet Relay Chat) channels can be used by attackers to get information from people more familiar with the code than themselves. Several of the authors of tools describe how to use their tools properly, but the vast majority writes their code as quickly as possible (Maximum Security, 2000). Some coders also make their programs extremely difficult to understand to keep the tools proliferation low and out of script kiddie hands (Fithen, 2000). On the other hand, the coder Fyodor has an extensive help and manual file, with examples of successful methods. Fyodor also has a newsletter

where he takes suggestions from his subscribers on the features that they want in future versions of nmap. One source of popular tools for this study was the results of a survey by Fyodor. Encompassing over 2000 security conscience users, they picked 50 of the most pertinent open source tools on the Internet, which is incorporated as Appendix A.

Many security sites provide the explanations and reasons why exploits work and how to determine if an attack has been made. Prospective attackers can view these reports and change their methods to remain stealthy. According to Maximum Security's anonymous author, a key for an attacker is to search the Usenet newsgroups for conversational threads on a particular exploit/weakness. At certain Usenet postings legitimate security personnel, open to be viewed by anyone subscribing to the newsgroup, will discuss problems or new details about vulnerabilities. One site named BUQTRAQ tracks a wide range of exploits. It is the communal meeting place to discover new software and tools. "BUQTRAQ is probably the Internet's most valuable resource for online reporting of UNIX-based vulnerabilities" (Maximum Security, 2000). One of the more interesting features is a listing of sites that are vulnerable to being used to help with DoS attacks (Sans, 2000). Originally designed to embarrass the sites into corrective action, this type of publicity gives attackers a ready list of resources.

Attackers have a wide range of resources to choose from to discover tools and vulnerabilities. In many cases the same warnings sent to sysadmins are notices of opportunity to potential attackers. Attackers get their information in the same way they let others write the code for tools: by using the labors of others. With a bit of research and time, a hacker can discover a significant amount of information on a new

vulnerability without having to invest the time to discover the knowledge himself or herself.

E. CONCLUSION

In conclusion, there is no one single "magic" tool that an attacker can use to break into any target. Specific tools take specific actions. However, by having knowledge of how to use tools, an attacker can follow a step-by-step process of gaining information to complete a successful attack. Owning the tools does not necessarily mean the attacker has the skill to know how to employ them. Dr. Longstaff of CMU CERT, recounts an attack he witnessed: "a script kiddie used an advanced break-in tool, but didn't know what to do once he was in – he was using DOS commands on a UNIX box" (2000), obviously the attacker was using the tool blindly. A large amount of knowledge is encoded into tools, but as of yet no tool does everything for the script kiddie.

The newest *successful* tools and techniques are constantly being updated on a large amount of websites. Potential attackers review the latest techniques from Internet experts, on both sides of the security fence, to find the best methods of attack. Online attack search engines for enumerated systems give attackers easy resources to specific tools that they wish to attack. Warnings and explanations from government organizations and software corporations give attackers information on ways to avoid detection and indirectly how to conduct successful attacks. The successful attacker using open source tools carefully chooses tools and methods by polling multiple sources, piggy backing on other peoples' experiences and expertise.

IV. FIVE APPLICATIONS OF CURRENT OPEN SOURCE TOOLS

This chapter presents five brief case studies that utilize open source tools found on the Internet today. Two are historical cases of recent attacks. The other three cases are notional possibilities that could occur using current open source tools. The goal of the chapter is to investigate why the cases were/could be successful and to examine the important aspects and considerations of each scenario.

The chapter is presented in two parts. Part one is a discussion on the two historical open source cases. Part two examines three theoretical applications of open source tools and some of their specific considerations. The chapter concludes with a small part on some of the considerations in using DDoS attacks.

A. HISTORICAL CASE STUDY ONE - *I Love You Worm*

The I Love You Worm (ILYW) was not an open source tool: it was a worm. At its core it was a *Visual Basic Program* that self-propagated through *Microsoft Outlook.* The ILYW is included in this chapter because it took advantage of an inherent weakness that is open source: a Microsoft feature that allowed the execution of code without safety precautions. Because it is written in Visual Basic Script (VBS), it only worked on Microsoft Windows applications such as *Microsoft Outlook.*

The worm traveled mainly by electronic mail. When the script was run, the worm would send copies of itself to every address in the Outlook file and alter several files on the affected system. The worm was successful because the standard features of Microsoft products were not safety conscious. Out of the box, *Microsoft Outlook* will run VBS

code that an email might have with it, even by just previewing the email. When a person viewed his email, the malicious code would infiltrate his system.

The ILYW was released around the first of May, 2000 and quickly spread around the world. According to CERT, at its height it affected about 500,000 individual systems and adversely affected many networks due to the high generated level of mail and file traffic (CERT Advisory, 2000). Interestingly, the ILYW also made its way onto four classified computer systems within the DOD infrastructure, most likely through the introduction of disks through personnel security lapses, rather than by any Internet connection (Plummer, 2000).

1. Important Aspects of the *I Love You Worm*

The ILYW demonstrates how a widespread application with an exploited weakness can have far reaching effects. If the author of the ILYW had exploited a weakness in the old *Borland Quattro Pro* application, the effects would have been obviously less intense. An application spread widely on the Internet with an exploited weakness, has more far-reaching consequences than a heterogeneous mixture of applications would. As shown in Figure 1, NT and Linux servers are the most popular servers on the Internet, and thus weaknesses in them would have the farthest reach (Netcraft, 2000).

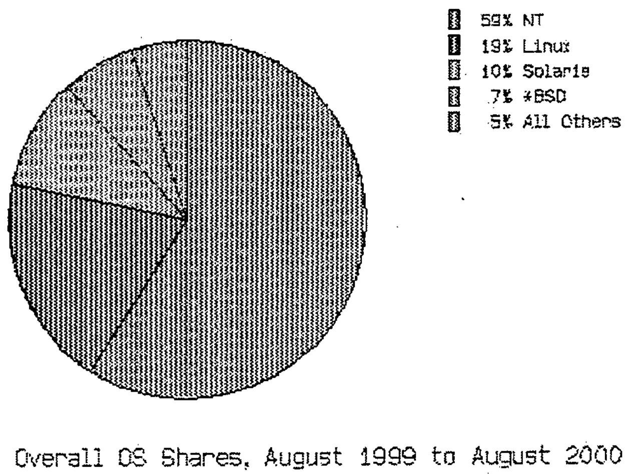

Figure 10. OS Server Distribution (From Ref. Attrition, 2000)

Since these OS's have the farthest reaches, they get the most attention in people's searches for weaknesses. It is a Catch-22: the most popular OS's with the farthest reaches will have the most vulnerabilities because they are the most popular. Those discovered vulnerabilities would have more impact because of the popularity of successful OS's. Figure 2 shows the number of vulnerabilities found per OS broken down in the last four years. As the chart shows, there is a sharp increase in vulnerabilities in the last two years.

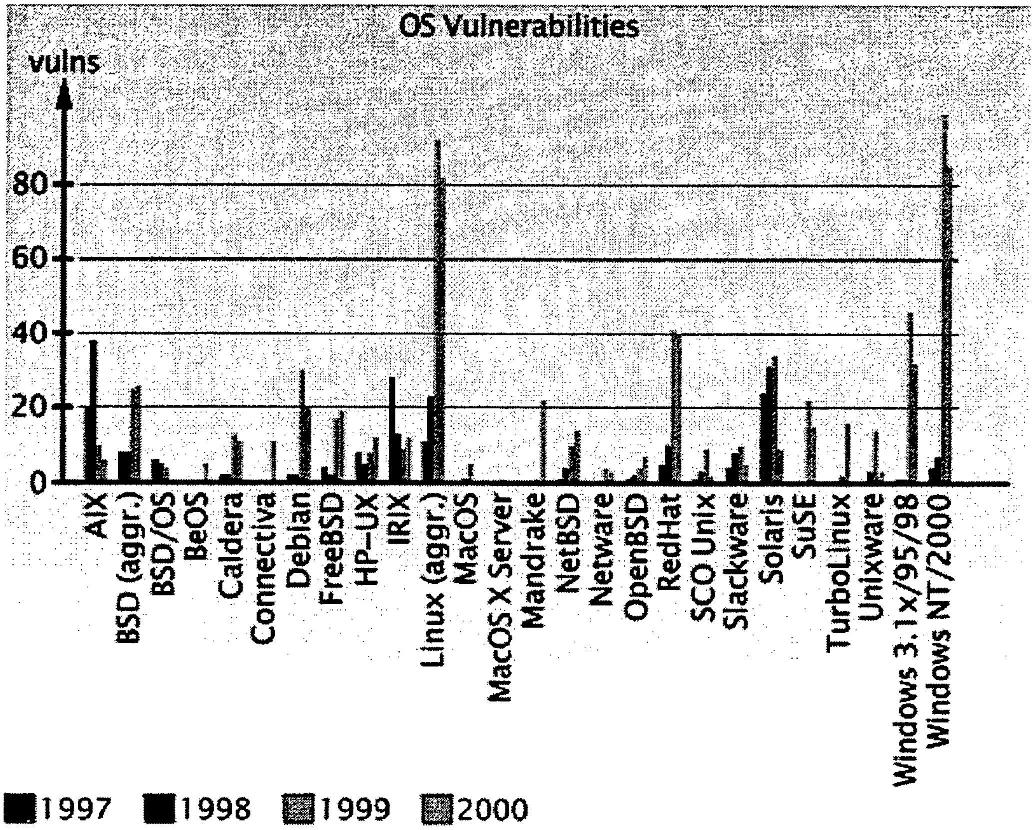

Figure 11. OS Vulnerabilities by Type (From Ref. Securityfocus, 2000)

Finally, many copycat worms were released in quick succession after the popularity of ILYW hit the mainstream press. All versions attempted to exploit the same weakness that the original ILYW did, sometimes improving on the "rookie application" code that was easily available. (Powell, 2000; Finley, 2000). They followed the "Vulnerability Life Cycle" examined in Chapter II, and had a much smaller effect than the original ILYW.

B. HISTORICAL CASE STUDY TWO – *Distributed Denial of Service*

> The attacks on Yahoo and other sites were pinpricks on the body of e-commerce and pretty insignificant overall. James Adams, CEO of Infrastructure Defense (Sheppard, 2000).

Over the course of several days in February 2000, a series of DDoS attacks hit several of the most popular sites on the Internet. Although effective only for a few hours at a time, the attacks instigated a widespread fear in many of not being able to protect themselves. These attacks were very notorious and produced a flurry of responses from all levels of society, from the press to the White House, as many felt that the base security of the Internet was at risk. The attacks were popular even to the point of momentarily affecting the performance of the stock exchanges: contributing to a 258-point slide in the DOW Jones average and a similar hit in the NASDAQ market (Pearlman, 2000).

The largest attack was on February 7th, when a DDoS matrix attacked Yahoo, a popular web portal. According to Global Center, Yahoo's ISP, they can handle up to 4.5 gigabits per second on their total network. On the day of the attack they received over 1 gigabits per second, effectively swamping the bandwidth that was set aside for Yahoo (Grice, 2000). Yahoo was forced to move to its back-up ISP to continue functioning, being forced offline for three hours.

The attacks continued throughout the week. On the 8th of February 2000, Buy.com, Ebay, CNN, and Amazon were hit, followed by ETrade and ZDNet on Wednesday. Overall, nine total sites were hit with all very similar DDoS attacks. The results varied from site to site: Yahoo had the highest bandwidth attack with 1 GPS while

Buy.com was hit the hardest with over 800 MPS, which was eight times its normal traffic.

When the FBI started investigating, they discovered that the same fifty IP addresses were involved in almost all of the attacks. However, according to Weld Pond of @Stake, "more likely hundreds of thousands of servers were used and the data forged to make it look like it came from only 50 addresses", in a form of IP spoofing, and part of a complex coordinated attack (Lemos, 2000). About two weeks after the other attacks, the investigators themselves were the victims of a denial of service; on Feb 18 www.fbi.gov was taken down for three hours in a DDoS attack. This underscored that "pretty much anyone is a target". (Dube, February 25, 2000). Included in the attack on the FBI was an attack on Microsoft. However, the DDoS didn't have enough bandwidth to be able to affect the Microsoft site.

The FBI believes that the DDoS attack tool, TFN2K or stracheldraht (German for "barbed wire"), was the software used for the attack. Several thousand computers were used in the attack, which points to a significant amount of time invested to break in and install the TFN2K code. The FBI felt that a Canadian teenager named "Mafiaboy" was the main culprit behind the denial of service attacks. Due to some IRC chat logs, the FBI was able to link him to having knowledge of unique aspects of the attacks and based on this and other evidence Mafiaboy was arrested and questioned last April. There has been much speculation as to whether Mafiaboy is the actual culprit, but just the fact that they seriously consider a 15-year-old boy a credible potential threat is interesting and troubling.

1. Important Aspects of the DDoS Attacks

The DDoS attacks were not a total surprise. CERT, along with other security related sites, had put out warnings and advisories the year before about the possibility of just such an attack. It wasn't a matter of "if", but of "when". Actually, the FBI had discovered in June, after the attacks, that many computers had been infiltrated with the daemon program for another DDoS attack (Thomas, 2000). It's possible that some computers that were in that report were used in the original February attack.

Mafiaboy, or whoever the attacker was, specifically targeted some university computers as part of his bandwidth source. University computers make great targets because of the high bandwidth universities usually lease, in addition to the centralized security policies. If one computer in a bank of computers at a university has a weakness, chances are that all the computers in the same room have the identical weakness.

In addition, according to Gary McGraw of Reliable Software technologies, "[Universities] don't always have the resources to guarantee good security. In fact, many are committed to openness, and that makes it very easy for someone to commandeer them" (CNET, 2000). Universities also have to balance the interests of their research with the need for security, and some security measures would hamper certain types of research. The specific targeting of education computers might give pause to the trend of putting a computer for each child in the public education. Right now, 95 percent of all schools have at least one computer, and these and future computers could be easy targets for attackers to get bandwidth (Krebs, 2000).

Finally, and most important, several companies that spent millions of dollars on security were vulnerable to open source tools that were freely available on the Internet. Although able to screen out massive amounts of packets, even the most expensive firewalls still need to spend a non-zero amount of time to examine the packets. If enough packets are sent, even ones obviously malformed or from a known hostile IP address, the firewalls can still be forced to slow down legitimate traffic. The companies were successfully attacked because the security weakness of others on the Internet were used against them.

C. THEORETICAL CASE STUDY ONE – TERRORISM ON THE CHEAP

One possible case that could occur involves a DDoS attack along the same lines as the Ebay and Yahoo attacks. But rather than one lone civilian, this case study involves a motivated, trained person who is sponsored by a terrorist organization/state. For this example, the terrorist state of choice is Iran because of its past history of terrorist support around the world, but almost any country could set up this operation (Arquilla/Shimeall, 2000). Iran, in this scenario has reason to visit harm upon the United States in the form of an Internet terrorist activity.

The scenario starts out with the goals of Iran: punishment of the United States for its activist foreign policy. The punishment, in this case, is the disruption of the American lifestyle by instilling uneasiness in the populace of the U.S. Iran wishes to break down the illusion that the US is safe from all enemies. Iran decides to send a homegrown computer specialist/agent into the United States to develop a computer attack capability. Iran makes a fake passport and identity for the man we will call "Bob".

Iran gives Bob $100,000 seed money for sustenance and to formulate his attack. He acquires twenty bank checking accounts throughout the geographic region he is located. An account charge card is issued for each account. Bob then puts $5,000 in each account. Using the charge card, Bob goes to all the local ISPs and secures several high-speed accounts, automatically drawing the payments from his charge card.

Bob could form a DDoS matrix using the open source tool TFN2K. Now, having several high-speed accounts throughout the region, a "master" computer will control each account. From there, he will start attacking computers from each master, using his skills to install the daemon software. He will spend the next few months amassing a bank of daemons for each master, laying low until he has a few thousand computers in his matrix.

Once the matrix is set up, Bob will have a wide range of computers to fulfill his needs. He can decide how much bandwidth and effect he wants by choosing the number of masters to use. This works well, because, while the authorities will eventually discover some masters and their daemons, Bob will still have his other masters to continue with the attacks. DDoS tools are a "wasting asset": every use after the first degrades the effectiveness of its next use (Arquilla, 2000). More and more of the matrix will be lost as computers are discovered as being helpers in the attacks. However, a one-time terrorist might not care that he will possibly lose some of his daemon computers as long as the intended attack succeeds once.

Bob has a whole range of activities that he can perform in order to disrupt society and gain a significant amount of press. Bob could have his daemon computers use an open source tool called 911 Worm. 911 Worm is an open source tool that polls the computers' ports for telephone lines and initiates a call to 911 in the area. The tool then

formats the hard drive, in hopes of erasing any sign of the intruding software. (CERT, 2000) Bob could use many computers in the same area to swamp the 911 circuits as a stand-alone activity or use it in conjunction with a conventional terrorist activity to increase the confusion and reaction time. The flooding would greatly hamper the response time causing more damage and press for the terrorists.

Additionally, the matrix could be used for denial of service attacks involving blackmail. In the DDoS attacks earlier in 2000, ETrade was one of the sites hit, effectively putting it out of commission for a few hours. Earlier in 1999, in an unrelated incident, ETrade was out for 22 hours which cost them five million dollars in revenue and 26 percent of their stock value, as well as damaging their reputation (Dube, 2000). The terrorists might be able to demonstrate a denial of service and then use this proof to blackmail ETrade or other financial sites into publishing their manifesto on their web sites. Even the public announcement of the intention of conducting a DDOS would affect the reputation and stock price of ETrade (McHugh, 2000).

Bob's range of activities is only limited by one's imagination. This is a very attractive scenario for Iran because of the high chance of success and the low chance of being caught before an attack is made. In addition, Iran would keep the attention of the American public as the attacks continue, despite the efforts of the FBI. Another important aspect is that Iran can deny culpability. Using the open source tools gives deniability, because of TFN2K's well-known signature, and since it is so widely available, anyone might be the culprit.

D. THEORETICAL CASE STUDY TWO – CHINA WITH AN ACE UP HER SLEEVE

China has remarked that if forced to fight, they will fight asymmetrically. During times of peace, China is interested in gaining access to the World Trade Organization and bettering U.S. economic ties. However, a patient and smart country such as China could be forming a hidden DDoS matrix for use in case hostilities between our two countries occur. There would be a great advantage to having such a structure in place once hostilities did erupt, in terms of time, speed, and security. Dan Kuehl, professor of the National Defense University, says Russia and China have the capability to attack the US power grid successfully and, in addition, "have clearly enunciated computer attack strategies aimed at sowing fear and crippling an adversary's military and commercial information infrastructure" (Loeb, 2000). China would keep this "ace up the sleeve" until its use was deemed necessary.

China surely has the required expertise to discover the significant nodes that would be damaging or sow significant confusion in the event of the matrix ever needing to be used. According to the Washington Post, over 95 percent of US military communications goes over civilian lines; these nodes might be especially enticing (Gray, 1997). One possible target could be the Satellite Communication (SATCOM) lines or their ground station transfers. China could lease a SATCOM line beforehand and study any weaknesses where an open source tool could have impact.

China could have computer attackers go out and make a bank of daemon computers throughout the world. The attackers could then pool all the IP addresses of the daemon computers into a central database to be controlled by a few master computers in

China or other safe places in the U.S. They could even form their own Dial-Up Bulletin Board Systems (BBS), which would not be connected to the Internet, to help their collection efforts. Their "agents", or undercover computer science students at US UC systems, could dial up and get the approved open source tools to infiltrate other systems.

Open source tools are important for this scenario for several reasons. First, the tools are widespread and have known signatures. Some infiltrated computers are bound to be discovered by their rightful owners and having a widespread open source tool gives a form of anonymity to the Chinese. In addition, the attack might be able to be pulled off without the target discovering that China was behind it. All the target would know is that it is getting hit by a notorious DDoS, and not necessarily the motivation or identity of the people behind the attack.

The DDoS attack could be combined with a military attack to compound the confusion and damage. The scenario would put China in a strategic dilemma however; China would no longer be anonymous and thus, would be attributable to the electronic attack. The military attack would have to be after the electronic attack for the maximum damage, thus a DDoS/military combination would forfeit some secrecy that a military attack is possibly underway. Finally, DDoS are not clear-cut weapons; there is always some spill over collateral damage to other countries (Longstaff, 2000). There are unintended consequences: China might hit people she didn't anticipate or possibly affect the Internet in ways more damaging to China than the United States.

E. THEORETICAL CASE STUDY THREE - GNUTELLA/NAPSTER TROJAN HORSES

The last case study involves taking advantage of the exploding popularity of Peer-to-Peer (P2P) connections. One of the consequences of the digital age is the ability to make perfect, compressed, digitally identical files, such as MP3 music files. Rather than purchasing a CD, one can find P2P software that allows you to find a digital copy on the Internet. Digital music files, and the ease with which to download them, have produced a fertile market for the development of software to share those files. Since the RIAA vs. Napster legal case started in 2000, the popularity of P2P connections has skyrocketed; it now has a 445 percent growth rate with over 20 million customers. (Barlow, October 2000, and Kelsey, September 2000).

P2P, in simple terms, forms a trusted relationship between two computers. A service like Napster, or software like Gnutella, gives users an opportunity to find and download files from other users' computers around the world, forming a logical link akin to a web page connection (Kuptz, 2000). In this regard, a user can bypass copyright payments and laws, and download files directly from another person's computer.

The weakness lies in the widespread use of P2P software and the possibility of downloading a Trojan Horse file that looks like a media file. There is an open source tool called "wrapster" that makes any file look like an MP3 file (The site is www.members.fortunecity.com/wrapster). An attacker could use wrapster in conjunction with Netbus or Sub7 RAT server files, joined by "jointer" as mentioned in Chapter Three, with an authentic music file to make a Trojan Horse music file. If the attacker then did this to a huge range of songs and allowed the P2P community access, his Trojan Horse

MP3 files would be spread to each computer that downloaded from him. When the song is played – the RAT server would be installed as well, forming a quick resource of accessible computers that could be found later.

F. DDoS CONSIDERATIONS

> There is essentially nothing a site can do with currently available technology to prevent becoming a victim of ... a coordinated network flood. The impact upon your site and operations is dictated by the (in)security of other sites and the ability of a remote attacker to implant the tools and subsequently to control and direct multiple systems worldwide to launch an attack (Distributed-Systems Intruder Tools Workshop, CERT Coordination Center, Nov. 1999).

Right now, an unknown number of attackers, sponsored or un-sponsored, could be amassing a huge "army" of computers. Similar to the buried clay armies of Xian, China, once enough bandwidth is found, their masters could have them rise out of the underground and attempt to wreck havoc. It is no coincidence that three of the case studies were DDoS attacks; having such a powerful tool with a lack of security against it makes it very enticing to use for the case studies. It is an attack known to have few defenses other than the effort needed to form it and the need for an individual to remain anonymous.

The best defense from a DDoS has been through the deterrence by punishment. Most solutions to the DDoS threat are involved with the aftermath and the prosecution of those responsible. In the case of the DDoS attacks earlier this year, the FBI has said, besides civil cases from the companies, the criminals, "face a five-year prison term for a first offense, 10 years if convicted of multiple and up to $250,000 per count" (Messner,

2000). The fear of a thorough search and administration of justice keep many potential offenders from initiating computer attacks (www.cybercrime.gov).

There are few solutions to a DDoS, such as filtering outflow IP addresses so they can't be spoofed, but then that only means those computers that started the DDOS will be located, the attack will still be performed (McHugh, 2000). For one-time attacks, like the China case, finding the daemon computers after the fact wouldn't matter. But for individual attackers or terrorist who wishes to use the matrix over and over again, the IP filtering would increase the "wasting asset" impact, as computers that took part in the attack would be located. Right now, there are few ISP's that egress filter IP addresses; thus, the security of one's site is partially impacted by the security of all other sites.

G. CONCLUSION

The five case studies described provide a small indication of what is possible with open source tools today. The theoretical cases could be over-shadowed by current events: "So far no one has died from a computer attack yet", but some possibilities definitely exist (Fithen, 2000). Country vs. country computer attacks have been envisioned, and practiced on the small scale in a few cases, but the appropriate responses to a true nation-wide attack have not been investigated thoroughly. (Office of General Counsel, May 1999).

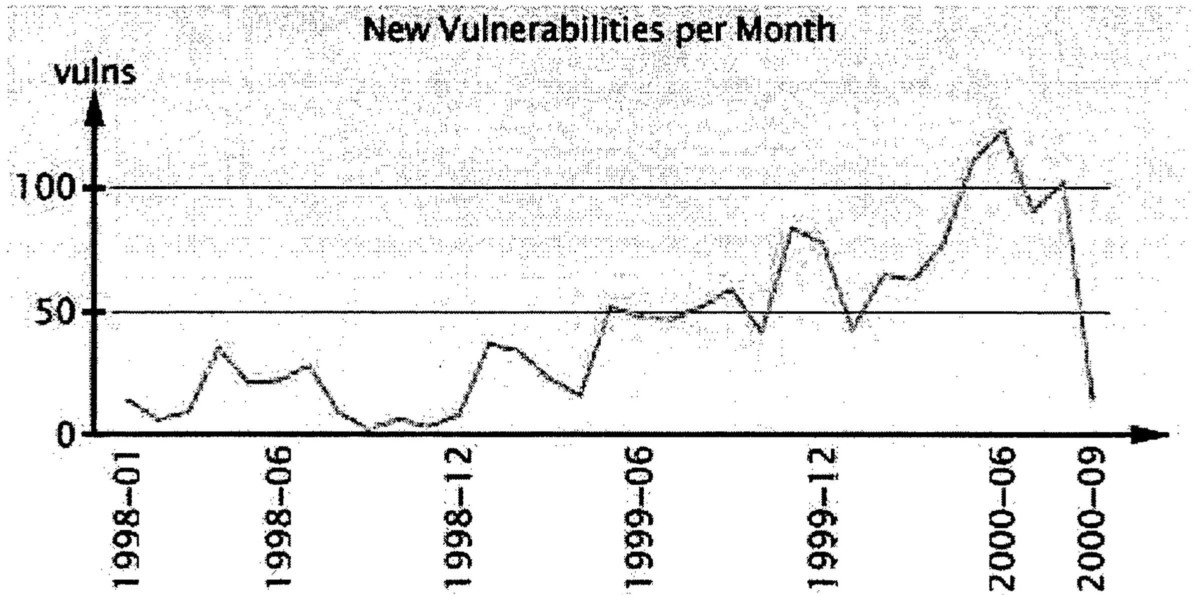

Figure 12. Vulnerability Trend (From Ref. Securityfocus, 2000)

Security is generally reacting to the threats and thus is always a step behind; therefore deterrence has remained the main solution to stop attacks. Figure 3 shows the recent trend for the discovery of vulnerabilities per month. Granted, not every one of these weaknesses is developed into an open source tool, however the knowledge of such weaknesses is a form of open source tool in itself. An attacker can still pick and choose of among the over than 50 new vulnerabilities each month, or nearly two a day, to find and attempt an exploit. It is important to note that the majority of these discovered vulnerabilities are immediately fixed by the vendors in the form of open source patches.

In conclusion, modern-day open source tools can have significant future impact and consequences due to the wide spread availability and the lack of real defenses (at least in the case of a DDoS attack). However, the five scenarios described might be considered extremes of what is possible; similar attacks are probably not more common due to the high probability of being discovered after the attempts, and the uncertain

nature of what nations might consider to be appropriate responses to a nation-to-nation Internet attack. Non-state attackers fall under the same considerations, as the unknown blowback might be more than they are willing to risk.

THIS PAGE INTENTIONALLY LEFT BLANK

V. FUTURE TRENDS

> You may advance and be absolutely irresistible, if you make for the enemy's weak points; you may retire and be safe from pursuit if your movements are more rapid than those of the enemy SunTzu, The Art of War.

A. INTRODUCTION

This final chapter consists of two parts: future trends of open source tools, and a summary of findings. The future trends section is derived from my observations and interviews over the last year of research. In it I have chosen seven trends that may bear upon developments in future open source tools. The summary of findings concludes this study with my "State of the Union" regarding the status of open source tools on the Internet today.

The trend of open source tools is captured in Figure 1. The chart, obtained from CERT, shows that, over time, a decreasingly sophisticated attacker can initiate increasingly sophisticated attacks. It appears that this trend will continue to progress in this fashion in the future as well. Additionally, the knowledge required to perform the whole range of attacks will decrease; in other words, as newer open source tools are developed, all attacks across the line will be easier to conduct than in the past, which would be represented in the chart as an overall lowering of the attacker's knowledge arrow.

However, as discussed in Chapter III, access to the tools does not guarantee their correct usage. The tools might provide entryways into a system, but in most cases, once in the system, do not provide the expertise to exploit the system. Expertise is still

needed to take full advantage of tools (Longstaff, 2000). Never the less, as described in Cyberterror, Prospects and Implications, "An unskilled attacker could [still] stumble upon a critical vulnerability [or tool] that produces substantial cascading effects" (Nelson, et al., 1999).

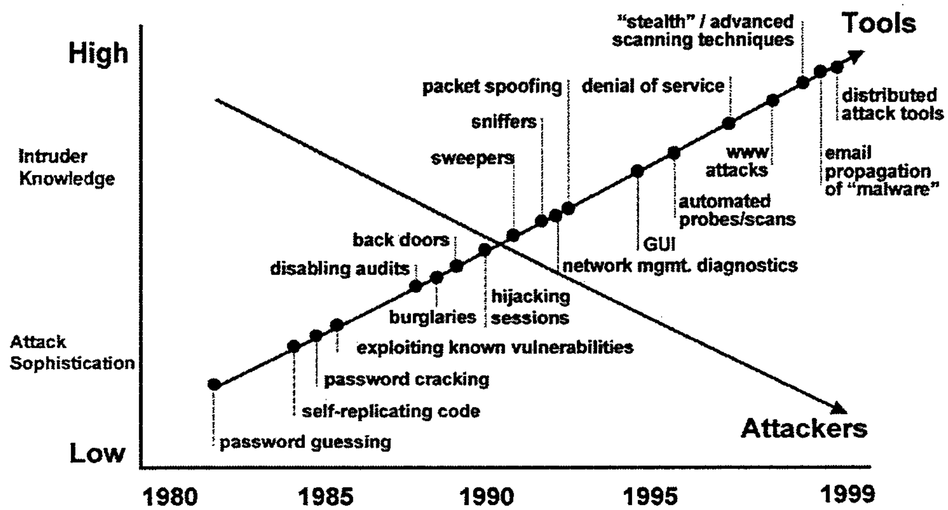

Figure 13. Attack Sophistication VS. Intruder Knowledge (From Ref. CERT, 2000)

B. TRENDS DISCOVERED DURING RESEARCH

Trend 1: Enhanced Multi-Tools

The first trend is the continued progression of the chart in Figure 1. In addition to more sophisticated operations being programmed, I think the next step is the combination of several different operations integrated into single tools, called multi-tools. This will have the effect of lowering the "barricades to entry" to attackers. While there are some

examples of this today (mainly in detection of vulnerabilities), the new tools will take the attacker through several steps of the attack sequence, ending with an option to conduct several forms of attack. The overall ease of use of tools will be enhanced, meaning less computer expertise needed, as almost all the operations will be done, "behind the scenes". Tools are also being developed with extensive help files, sometimes incorporated into the program themselves, providing instruction on how to properly use the tool under different conditions.

The multi-tools will have more automation in the set-up for an attack and the attack itself. More programmer knowledge will be coded into scanning, enumerating and providing direction on the likely avenues to pursue a successful attack. They will be similar to the multi-vulnerability detection software on the market and Internet today: able to determine if systems are vulnerable by testing a wide range of known vulnerability operations. The Security Administrator's Tool for Analyzing Networks (SATAN), an older, popular vulnerability testing program mainly used on UNIX based computers, started out and continues to be used in legitimate ways; however, it is also used to test other peoples' systems for multiple vulnerabilities. Attackers will continue to use legitimate tools in illegal ways.

Another aspect of the trend is consolidation of the response databases. More scanning responses will be incorporated into multi-tools. There might be a combination of a wide range of OS, firewall, router and platform responses, like the nmap database, joined with a range of exploits that those targets might have. Additionally, there are scripts called "rootkits" that help get root level access and hide the fact that attackers were ever there. Rootkits are specific to the target OS and software. They are powerful

tools that exceed the skill level of the average sysadmin (Dittrich, 2000). I think programmers are going to combine a range of rootkits into one larger rootkit database which will be utilized to attack a variety of computer systems. A wider reaching rootkit database would be more effective and efficient, since it would target more systems and consolidate the past work of other programmers.

Overall, future multi-tools will take the current situation a step further by becoming "one-stop shopping" tools for a select set of systems. Multi-tools will determine the vulnerabilities, then give options of a variety of attacks that could be used against the target, all in a GUI environment. An attacker would only have to set the range of IP addresses to search, and the tool would basically do the rest, prompting you with options to continue or stop.

Finally, programmers might code a "framework" program that attackers would be able to fill with open source modules that perform specific parts of an attack. For example, one would be able to load different methods from different programmers and be able to execute them from a single tool. The framework would be the epitome of an open source tool and benefit from the coding of programmers around the globe. Much like the ability to add filters to *Adobe Photoshop* and the downloadable signatures from McAfree and Norton, an attacker would be able to continually upgrade as new methods are coded into modules that would add features and techniques into this new "open source" coded multi-attack tool.

Trend 2: Explosion of Internet Capable Devices

The Internet is undergoing explosive growth. As of 10 July 2000, there were two billion unique pages on the Internet, "and that number is growing to the tune of 7 million pages a day" (McGuire, 2000). By 2001, according to Cyveillance.com, the Internet will have doubled in size to 4 billion unique pages. The increasing popularity of the Internet provides a huge market incentive to develop Internet devices capable of tapping into this vast amount of data and the inter-communication possibilities.

A new developing trend is the ability to attach the Internet to everything possible, from refrigerators to hand-held PDAs. A project called Bluetooth, backed by over 2000 of the largest Internet companies, is the most likely future standard for wireless communication between the Internet devices. The project site (www.bluetooth.com) envisions auto-synchronization and seamless transfer of bits between any number of Bluetooth capable applications. The trend would be helped by the "Internet on a chip" – a quick option to connect the device to the Internet with its own IP address.

Your home, workplace, and car will all be logically connected and able to talk to each other. In addition, similar devices will automatically connect for immediate communication: the instant your milk goes bad, voice messages can be sent to wherever you are, giving you the option of ordering some more, or having your refrigerator do it. One would have a "personal network" to help with a variety of needs and wishes. (Hibbard, 2000).

Unfortunately, open source tools would take advantage of the trusting relationship between the multiple devices. Safeguarding all of the devices would be

difficult, especially for private citizens. End users with only one processor and connection have a hard time now; but with five or more connections, more chances of being vulnerable to an attack increase dramatically. New attack tools could query more of a target's computers. Once attacked or infected, attackers could hijack your bandwidth or be able to track your transactions, habits, and financial status.

Finally, one consequence of the proliferation of Internet capable devices is the shrinking availability of IP addresses. Each new device would require its own IP address. Right now the Internet is working with IPv4, with a theoretical limit of 4,294,967,296 unique IP addresses, based upon a 64-bit address system. To continue the expansion of the Internet, more IP addresses have to be made available. IPv6 is the newest proposal to expand the number of addresses and it is based upon a 128-bit address system, 4 billion times bigger than the current IPv4 system.

Incorporating IPv6 creates new concerns for open source tools and their developers. Along with expanding the number of addresses, IPv6 was developed to upgrade the old packet architecture and incorporate some helpful security measures. One of the key features of IPv6 is its reverse IP address checking capability. This means IP addresses will no longer be able to be spoofed as easily, which would have a deterrent effect on some possible attackers. In addition, IPv6 employs some encryption bits in the header to allow verification that the packet has not been altered. (Bay network and Nokia research center, 1999). This is also currently being achieved with systems using IPsec as well.

Trend 3: Wireless Networks - The Absence of an "Air Gap"

Another trend of the future that uses the explosion of Internet devices is the wireless "revolution", taking place in the workplace, home, and on the body. Similar to third world countries installing cell phone networks, wireless connectivity allows the user to leapfrog the hardwire infrastructure requirement that would usually go along with the bandwidth. However, wireless communication erases the security-enhancing "air gap" on systems. Although wireless systems could be highly mobile, not needing a physical connection is a benefit that could be turned into a liability or vulnerability. Instead of being able to physically remove any possible connection to the internet, hardware will have wireless receivers that can be taken advantage of. These machines will be "always connected" unless specifically taken out of the receiving mode. Open source tools could be developed to exploit these capabilities, possibly by a blind transmission of malicious software to wireless machines.

Additionally, wireless systems like Bluetooth or other systems utilizing a Wireless Local Loop (WLL) could be vulnerable to many more forms of attack besides software manipulation, such as electronic warfare jamming, burnout, and hijacking. As the saying goes, "a chain is only as strong as the weakest link". Adding extra links to a WLL might create extra vulnerabilities that give newly developed open source tools the ability to exploit.

Trend 4: High Bandwidth Proliferation

> The increasingly pervasive, high-speed, always on connection of DSL and cable have provided the foundation to build [the new Internet] upon. As a result, the Yankee Group predicts that home networks will mushroom from some 650,000 in existence today to more than 10 million by 2003 (Clyman, 2000).

There are over 144 million Americans surfing the net at home, which is 52 percent of the population (Kelsey, 2000). The US Internet population has a 35 percent growth rate and there will be 10 million people employed in the Internet economy by 2002, 5.8 million of them in the U.S. (Krebs, 2000). Needless to say, there is a huge market incentive for high speed Internet access.

Demand for high-speed access is propelling DSL and cable services as the successors of dial-up modem service. Consumers demand faster and faster connections and DSL/cable hook-ups are the new broadband suppliers. Traditionally hampered by hardware requirements, new research has allowed software to handle some of the old hardware concerns, opening DSL to huge potential markets (Gartner, 2000).

DSL and cable connections make a target more vulnerable to attack: they are connected to the Internet on a persistent basis and are not phone-line based. Thus, DSLs usually have a constant IP address as long as the computers are powered. An attacker can target a persistent IP address easier than a modem dial-up, since it is harder to anticipate when a dial-up will be connected. A persistent IP address permits an attacker time to test the whole range of tools, exploits, and techniques until one works (Konda, 2000). Additionally, since these are home networks, they are usually more vulnerable than a

Additionally, since these are home networks, they are usually more vulnerable than a high bandwidth connection at a business or organization that might have a security manager.

According to Frank Prince of Forrester Research,

> [An attacker] would have to take over 10,000 of [DSL] computers instead of 500 large servers so it means more work for them, but I don't doubt that we'll see it. Sooner or later, there is no question that someone will have marshalled a large number of private computers to be used in a high-profile attack (Manjoo, 2000).

As new open source tools become available for this new popular service, the security of home networks might not keep up and actually be more vulnerable than a dial-up service. Another consideration is that open source tools might be utilized for a longer time because it would still be worth the effort for an attacker to get the higher speed access even if he gets fewer high value targets.

Trend 5: Common Enumeration of Vulnerabilities

The last trend is the formation of a common database of vulnerabilities by several different organizations. It is thought that the function of such a database would have many beneficial attributes to combat the effects of vulnerabilities and tools that exploit weaknesses. The common database would help in the exchange, interpretation, and correlation of information that would shorten the effectiveness or even neutralize emerging vulnerabilities (Baker, 1999).

However, the formation of a common database has three main obstacles (Meunier, 1999). The first obstacle is the technical aspect of sharing data. The proper nomenclature, the format of the data, the correctness of the data, and even the language used to share the database are difficult to agree upon. Additionally the trustworthiness of the data from outside sources might deter one organization from using data that was compiled by a suspect source.

Secondly, there is a motivational obstacle to overcome. Compiling the data is expensive and time consuming, and dividing the process equally is difficult when it would be to one organization's benefit to be a "free rider". Conversely, it would also benefit an organization to withhold information from their competitors. Lastly, the consequence of sharing a database might provide information that could be used in an attack against a target, prompting the possibility of a lawsuit. Actually, the Department of Defense (DOD) will have a common database of vulnerabilities and tools by early 2001 (Seffers, 2000). However, the government has a policy of not sharing its database it has due to the legal restraints of it having been formed through taxpayer's money.

Overall, a database of vulnerabilities could have a dampening effect on the longevity of open source tools. If the database was updated frequently and distributed without charge, a new vulnerability could get widespread attention before a tool could be produced to take advantage of it. Additionally, there would also be a decrease in the amount of redundant effort in the security community, allowing more cooperation in the security community.

Trend 6: Elevated Status of Programmers

Hackers, crackers, and attackers within the digital community get a fair amount of press and notoriety. There is substantial fame associated with some of the more popular hackers: almost every one in the computer industry has at least heard of Mitnick, Poulsen, Mafiaboy, Morris, Mixter, Rain Forest Puppy, and Fyodor. At DEFCON 2000, a few hackers were introduced as heroes, and certainly a subculture of fame has developed around them. For whatever reason these hackers and crackers had initially started to form code, the hacking community has swept them up and given them encouragement to continue in their development. Much like the story of the man who got a job at a bank after robbing them, by showing how he did it, hackers such as Mitnick and Poulsen have used their fame to gain employment. New hackers have a motivation to make a name for themselves: respect, notoriety, self-esteem, and employment.

Additionally, hackers such as Fyodor want to please their fans. Fyodor, author of nmap, has sent out surveys to members of his hacking subscriber circle asking for features they wish to see in future versions of nmap. What's more, Blade, author of jointer and tHing, has also shown reaction to his followers in his web site. The new popularity is motivating the programmers to keep their followers by improving their products, and actively seeking out their input. Fear outside the community provides press and fame, while admiration from their fans provides respect, a well-visited web site, and suggestions for new features.

Overall, the attacker community is becoming better organized. Several sites, such as www.rootkit.com, openly encourage programmers to join their team and upload their

software. With a small pool of expert hacking elite, there is an atmosphere of a "Joint-Combined" team effort to hack some security vulnerabilities discovered. The weeding out process ensures that successful tools are widely distributed while faulty tools die on the vine. Finally, there seems to be an increasing interaction on all levels of communication between attackers, from popular conventions such as Black Hat and DEFCON, to online web sites and newsgroup subscriptions, fueling more opportunities for the development of higher quality tools.

Trend Seven: DDoS

> But if someone maliciously takes down the biggest nodes, you can harm the system in incredible ways...The bad news is that Internet terrorists could cause great damage by targeting the most connected router (Reuters, 2000).

One of the leading experts on Distribute Denial of Service attacks is Dr. David Dittrich. He is a professor at Washington University and has produced studies of each of the major DDoS tools on the Internet. In an interview in the online magazine SlashDot, Mr. Dittrich outlines a few of the functions he feels that future DDoS tools will incorporate. For example, one of the functions he feels that could enhance the tfn2k tool is encryption of the communication between the masters and the daemons. He also thinks "it won't be long before someone *tries* to take that next step and further automate the process of scanning & intrusion to constitute DDoS networks." (Roblimo, 2000).

DDoS could be enhanced by the use of new multi-tools to self propagate. They could automatically scan a random range of IP addresses until the responses indicate a system on the other end that contains vulnerabilities that the multi-tool is able to exploit. A master computer in the matrix might only be able to "infect" one type of system, while other masters are specialized in other systems, or the whole DDoS matrix could specifically target only one type of system. Once set into motion, a massive amount of computers could be theoretically harnessed with very little effort. An automatic self-propagating matrix has a whole range of inherent problems dealing with growth restraint, communication, and coordination for an attack making it unreasonable for the time being (Shimeall, 2000). However, if a tool restricted itself to a short growth cycle, possibly with a time-initiated attack schedule not requiring communication with a master, those problems could be overcome.

C. FUTURE APPLICATIONS OF DISTRIBUTED OPEN SOURCE TOOLS/RATs

> Information systems are so vital to the [US] military and civilian society that they can be the main targets in war, and they can also serve as the main means for conducting offensive operations (Arquilla, 1999).

DDoS is a waste of bandwidth and processor ability; it is a simple attack that is very "loud" and gets a lot of attention, making it a wasting asset. In a DDoS, the bandwidth and processors are used to form packets as quickly as possible. Instead of a simple attack, new distributed tools will utilize the matrix computers in different ways. The two major resources in a distributed matrix that can be utilized are bandwidth and

processor cycles. Additionally, the logical positions of the nodes in the matrix could also be used to advantage.

One alternate use of a distributed net is being performed by the SETI project. SETI, the Search for Extraterrestrial Intelligence, has a voluntary distributed network that utilizes idle processor time to decipher signal patterns from outer space. They have formed the most powerful computer in history using an average online number of 500,000 computers in their matrix, calculating close to eight teraflops per day (Patrizo, 2000). Similar distributed networks such as COSM and the Globus Project also harness large numbers of computers for idle processor time.

A tool that allows the user to decide what is to be computed would permit an attacker to utilize such networks in several ways. Of course, determining the opportune time to engage the processor of an unknowing target, and still remain hidden, would be difficult. A distributed network has been used to break the DES algorithm used to encrypt files, involving over 100,000 computers taking 22 hours (http://www.eff.org/descracker). An attacker could gather intelligence from a target, using an open source sniffer, and utilize a matrix to help crunch user passwords or even decrypt messages. Granted, a large amount of time would be needed for decryption, but the encryption would still be eventually broken, much faster than by a single computer. Sensitive information, that is not *time dependent*, would eventually be discovered.

An attacker might also be able to use a matrix to gather intelligence or damage the reputation of a target. If a target system is unable to be attacked, possibly the computers and routers "around" it could be. An attacker might be able to infect computers and routers that are between the target and another computer. A percentage of packets that

originate from the target might be able to be copied from surrounding infected systems to another site for analysis. According to Nielsen-Net ratings, the average unique sites visited per month are ten; the key is to find a machine that is in between the target and its destination. With a large enough matrix, an attacker could determine information about the target without infecting the target itself, but by its communication through and with surrounding systems.

D. FINDINGS

> The amount of work required to secure your system is directly proportional to the value you place on your server (Anonymous).

Advances are being made to make the Internet more secure. IPsec, which allows a Virtual Private Network connection on IPv4 and IPv6, is a step toward a more secure stance. Additionally, the MITRE corporation has produced a dictionary of vulnerabilities designed to shorten the period of effectiveness for vulnerabilities. However, computer systems will always remain vulnerable. There will always be possible exploitations in things made in a market driven economy that values speed and ease of use over security (Dittrich, 2000). The number of new applications and systems, along with all the possible connections between them is rapidly increasing. No company or organization, not even Microsoft with over 40,000 employees, has all the possible expertise in the world in computer programming or hardware – there will always be more expertise outside than inside a product. Vulnerabilities will always be found in new and existing applications, mainly if a portion of the populace is motivated to find vulnerabilities. It is almost impossible for a company to test completely all the parameters of a released product, and

vulnerabilities will be found in ways a company couldn't prognosticate. To exploit these vulnerabilities, experts will continue to program scripts, for whatever reason, that will eventually find their way into being developed as open source tools.

Once a vulnerability is discovered and a tool is written, the effectiveness of the tool follows a life cycle pattern. Initially effective, as the tool gains notoriety, its own fame works against itself and defenses are developed to counter its effectiveness. However, a vulnerability is never completely blocked and could have resurgence due to the expanding pool of new systems and as new vulnerabilities take center stage.

Attackers have a wide range of ways to attack using open source tools. They also have a variety of ways to find their targets. They can search methodically to find a specific organization or perform a wide sweep for systems responding to a vulnerability. Open source tools help in the search and attacking of computer systems. Sophisticated attacks still require skill, however open source tools carry some expertise of their coder inherently, which makes systems vulnerable to attackers that otherwise wouldn't be able to threaten it.

Open source tools are easy to find on the Internet. There are a multitude of commercially sponsored, professional sites that compete to provide the best and most recent tools and vulnerability announcements. A potential attacker can also follow the advisories of security organizations to get a good idea of which tools would be currently effective on systems that might not be security conscious. Effective open source tools are widely advertised and gain quick notoriety through the Internet; several sites even have "vulnerabilities of the day" (www.securityfocus.com).

There are hundreds of new vulnerabilities discovered every year. Accordingly, the number of attacks has also increased every year. (CERT, 2000). Less and less ability is needed to use them effectively The more popular the application, the more probable that a vulnerability will be discovered due to the higher percentage of people searching for a weakness. As the Internet grows, so will the threats, and more and more people will become attackers or unknowing helpers of attackers through their own vulnerabilities.

Open source tools are the resumes of a new breed of popular figures. The motivation of fame and reputation keep people toiling to exploit weaknesses and develop tools. Additionally, there appears to be a trend of substantial testing of the open source tool before its release – in order to protect their reputation and please their "fans". Quality products are demanded and certain programmers have a reputation of putting out quality tools.

Open source tools will have an increasing impact on the security of the Internet due to its fantastic growth and the fact that the "emphasis on security is almost always on the end user" (Manjoo, 2000). As more and more of people's lives are wired into the Internet, more and more security vulnerabilities will exist. Open source tools will always be with us as long as the Internet continues its fantastic growth. As Mr. Dittrich notes, there are over 21 million new hosts every month. But more importantly, there are not 21 million new sysadmins every month – and the difference between the two is very important to the security of the Internet and the effectiveness of open source tools.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A
Top 50 Security Tools

The following is the text version of a web site at www.insecure.org outlining the survey conducted by Fyodor:

In May/June of 2000, we conducted a survey of 1200 Nmap users from the nmap-hackers mailing list to determine their favorite security tools. Each respondant could list up to 5.

I was so impressed by the list they created that I am putting the top 50 up here where everyone can benefit from them. I think anyone in the security field would be well advised to go over the list and investigate any tools they are unfamiliar with. I also plan to point newbies to this page whenever they write me saying "I do not know where to start".

Respondents were allowed to list open source or commercial tools on any platform. Commercial tools are noted as such in the list below.

I may change this list occasionally as new tools are created and others fade into obscurity due to security enhancements becoming mainstream. Or maybe I'll just have another survey next year.

Also note that many of the descriptions in this list were taken from the Debian package descriptions, the Freshmeat descriptions, or from the home pages of the application. I didn't count any votes for Nmap because the survey was taken on an Nmap mailing list.

Without further ado, here is the list (starting with the most popular):

Nessus http://www.nessus.org/

Description: Remote network security auditor, the client The Nessus Security Scanner is a security auditing tool. It makes possible to test security modules in an attempt to find vulnerable spots that should be fixed. . It is made up of two parts: a server, and a client. The server/daemon, nessusd, is in charge of the attacks, whereas the client, nessus, interferes with the user through nice X11/GTK+ interface. . This package contains the GTK+ 1.2 client, which exists in other forms and on other platforms, too.

Netcat http://www.l0pht.com/~weld/netcat/

Note: This is an unofficial site
Description: TCP/IP Swiss army knife A simple Unix utility which reads and writes data across network connections using TCP or UDP protocol. It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts. At the same time it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities.

Tcpdump http://www.tcpdump.org/

Description: A powerful tool for network monitoring and data acquisition This program allows you to dump the traffic on a network. It can be used to print out the headers of packets on a network interface that matches a given expression. You can use this tool to track down network problems, to detect "ping attacks" or to monitor the network activities.

Snort http://www.snort.org/

Description: flexible packet sniffer/logger that detects attacks Snort is a libpcap-based packet sniffer/logger which can be used as a lightweight network intrusion detection system. It features rules based logging and can perform content searching/matching in addition to being used to detect a variety of other attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and much more. Snort has a real-time alerting capability, with alerts being sent to syslog, a separate "alert" file, or even to a Windows computer via Samba.

Saint http://www.wwdsi.com/saint/

Description: SAINT (Security Administrator's Integrated Network Tool) is a security assessment tool based on SATAN. Features include scanning through a firewall, updated security checks from CERT & CIAC bulletins, 4 levels of severity (red, yellow, brown, & green) and a feature rich HTML interface.

Ethereal http://ethereal.zing.org/

Description: Network traffic analyzer Ethereal is a network traffic analyzer, or "sniffer", for Unix and Unix-like operating systems. It uses GTK+, a graphical user interface library, and libpcap, a packet capture and filtering library.

Whisker http://www.wiretrip.net/rfp/p/doc.asp?id=21&iface=2

Description: Rain.Forest.Puppy's CGI vulnerability scanner

Internet Security Scanner http://www.iss.net/

Note: This tool costs significant $$$ to use, and does not come with source code.
Description: A popular commercial network security scanner.

Abacus Portsentry http://www.psionic.com/abacus/portsentry/

Description: Portscan detection daemon PortSentry has the ability to detect portscans(including stealth scans) on the network interfaces of your machine. Upon alarm it can block the attacker via hosts.deny, dropped route or firewall rule. It is part of the Abacus program suite. . Note: If you have no idea what a port/stealth scan is, I'd recommend to have a look at http://www.psionic.com/abacus/portsentry/ before installing this package. Otherwise you might easily block hosts you'd better not(e.g. your NFS-server, name-server, ...).

DSniff http://naughty.monkey.org/~dugsong/dsniff/

Description: A suite of powerful for sniffing networks for passwords and other information. Includes sophisticated techniques for defeating the "protection" of network switchers.

Tripwire http://www.tripwire.com/

Note: Depending on usage, this tool may have expensive licensing fees associated with it.
Description: A file and directory integrity checker. Tripwire is a tool that aids system administrators and users in monitoring a designated set of files for any changes. Used with system files on a regular (e.g., daily) basis, Tripwire can notify system administrators of corrupted or tampered files, so damage control measures can be taken in a timely manner.

Cybercop Scanner http://www.pgp.com/asp_set/products/tns/ccscanner_intro.asp

Note: This tool costs significant $$$ to use, and does not come with source code. A powerful demo version is available for testing.
Description: Another popular commercial scanner

Hping2 http://www.kyuzz.org/antirez/hping/

Description: hping2 is a network tool able to send custom ICMP/UDP/TCP packets and to display target replies like ping does with ICMP replies. It handles fragmentation and arbitrary packet body and size, and can be used to transfer files under supported protocols. Using hping2, you can: test firewall rules, perform [spoofed] port scanning, test net performance using different protocols, packet size, TOS (type of service), and fragmentation, do path MTU discovery, tranfer files (even between really Fascist firewall rules), perform traceroute-like actions under different protocols, fingerprint remote OSs, audit a TCP/IP stack, etc. hping2 is a good tool for learning TCP/IP.

SARA http://www-arc.com/sara/

Description: The Security Auditor's Research Assistant (SARA) is a third generation security analysis tool that is based on the SATAN model which is covered by the GNU GPL-like open license. It is fostering a collaborative environment and is updated periodically to address latest threats.

Sniffit http://reptile.rug.ac.be/~coder/sniffit/sniffit.html

Description: packet sniffer and monitoring tool sniffit is a packet sniffer for TCP/UDP/ICMP packets. sniffit is able to give you very detailed technical info on these packets (SEC, ACK, TTL, Window, ...) but also packet contents in different formats (hex or plain text, etc.).

SATAN http://www.fish.com/satan/

Description: Security Auditing Tool for Analysing Networks This is a powerful tool for analyzing networks for vulnerabilities created for sysadmins that cannot keep a constant look at bugtraq, rootshell and the like.

IPFilter http://coombs.anu.edu.au/ipfilter/

Description: IP Filter is a TCP/IP packet filter, suitable for use in a firewall environment. To use, it can either be used as a loadable kernel module or incorporated into your UNIX kernel; use as a loadable kernel module where possible is highly recommended. Scripts are provided to install and patch system files, as required.

---

iptables/netfilter/ipchains/ipfwadm http://netfilter.kernelnotes.org/

Description: IP packet filter administration for 2.4.X kernels Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. The iptables tool also supports configuration of dynamic and static network address translation.

Firewalk http://www.packetfactory.net/Projects/Firewalk/

Description: Firewalking is a technique developed by MDS and DHG that employs traceroute-like techniques to analyze IP packet responses to determine gateway ACL filters and map networks. Firewalk the tool employs the technique to determine the filter rules in place on a packet forwarding device. The newest version of the tool, firewalk/GTK introduces the option of using a graphical interface and a few bug fixes.

Strobe http://www.insecure.org/nmap/index.html#other

Description: A "Classic" high-speed TCP port scanner

L0pht Crack http://www.l0pht.com/l0phtcrack/

Note: No source code is included (except in research version) and there is a $100 registration fee.
Description: L0phtCrack is an NT password auditing tool. It will compute NT user passwords from the cryptographic hashes that are stored by the NT operation system. L0phtcrack can obtain the hashes through many sources (file, network sniffing, registry, etc) and it has numerous methods of generating password guesses (dictionary, brute force, etc).

John The Ripper http://www.openwall.com/john/

Description: An active password cracking tool john, normally called john the ripper, is a tool to find weak passwords of your users.

Hunt http://www.cri.cz/kra/index.html#HUNT

Description: Advanced packet sniffer and connection intrusion. Hunt is a program for intruding into a connection, watching it and resetting it. . Note that hunt is operating on Ethernet and is best used for connections which can be watched through it. However, it is possible to do something even for hosts on another segments or hosts that are on switched ports.

OpenSSH / SSH http://www.openssh.com/
http://www.ssh.com/commerce/index.html

Note: The ssh.com version cost money for some uses, but source code is available.
Description: Secure rlogin/rsh/rcp replacement (OpenSSH) OpenSSH is derived from OpenBSD's version of ssh, which was in turn derived from ssh code from before the time when ssh's license was changed to be non-free. Ssh (Secure Shell) is a program for logging into a remote machine and for executing commands on a remote machine. It provides secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel. It is intended as a replacement for rlogin, rsh and rcp, and can be used to provide rdist, and rsync with a secure communication channel.

---

tcp wrappers ftp://ftp.porcupine.org/pub/security/index.html

Description: Wietse Venema's TCP wrappers library Wietse Venema's network logger, also known as TCPD or LOG_TCP. . These programs log the client host name of incoming telnet, ftp, rsh, rlogin, finger etc. requests. Security options are: access control per host, domain and/or service; detection of host name spoofing or host address spoofing; booby traps to implement an early-warning system.

Ntop http://www.ntop.org/

Description: display network usage in top-like format ntop is a Network Top program. It displays a summary of network usage by machines on your network in a format reminicent of the unix top utility. . It can also be run in web mode, which allows the display to be browsed with a web browser.

---

traceroute/ping/telnet http://www.linux.com/

Description: These are utilities that virtually all UNIX boxes already have. In fact, even Windows NT has them (but the traceroute command is called tracert).

NAT (NetBIOS Auditing Tool) http://www.tux.org/pub/security/secnet/tools/nat10/

Note: This is an unofficial download site.
Description: The NetBIOS Auditing Tool (NAT) is designed to explorethe NETBIOS file-sharing services offered by the target system. It implements a stepwise approach to gather information and attempt to obtain file system-level access as though it were a legitimate local client.

scanlogd http://www.openwall.com/scanlogd/

Description: A portscan detecting tool Scanlogd is a daemon written by Solar Designer to detect portscan attacks on your maschine.

Sam Spade http://samspade.org/t/
http://www.samspade.org/

Description: Online tools for investigating IP addresses and tracking down spammers.

NFR http://www.nfr.com/

Note: Source code was once freely available but I do not know if this is still the case. Some usage may cost money.
Description: A commercial sniffing application for creating intrusion detection systems. Source code was at one time available, but I do not know if that is still the case.

---

logcheck http://www.psionic.com/abacus/logcheck/

Description: Mails anomalies in the system logfiles to the administrator Logcheck is part of the Abacus Project of security tools. It is a program created to help in the processing of UNIX system logfiles generated by the various Abacus Project tools, system daemons, Wietse Venema's TCP Wrapper and Log Daemon packages, and the Firewall Toolkit© by Trusted Information Systems Inc.(TIS). . Logcheck helps spot problems and security violations in your logfiles automatically and will send the results to you in e-mail. This program is free to use at any site. Please read the disclaimer before you use any of this software.

Perl http://www.perl.org/

Description: A very powerful scripting language which is often used to create "exploits" for the purpose of verifying security vulnerabilities. Of course, it is also used for all sorts of other things.

Ngrep http://www.packetfactory.net/Projects/ngrep/

Description: grep for network traffic ngrep strives to provide most of GNU grep's common features, applying them to the network layer. ngrep is a pcap-aware tool that will allow you to specify extended regular expressions to match against data payloads of packets. It currently recognizes TCP, UDP and ICMP across Ethernet, PPP, SLIP and null interfaces, and understands bpf filter logic in the same fashion as more common packet sniffing tools, such as tcpdump and snoop.

Cheops http://www.marko.net/cheops/

Description: A GTK based network "swiss-army-knife" Cheops gives a simple interface to most network utilities, maps local or remote networks and can show OS types of the machines on the network.

Vetescan http://www.self-evident.com/

Description: Vetescan is a bulk vulnerability scanner which contains programs to check for and/or exploit many remote network security exploits that are known for Windows or UNIX. It includes various programs for doing different kinds of scanning. Fixes for vulnerablities are included along with the exploits.

Retina http://www.eeye.com/html/Products/Retina.html

Note: Commercial product with no source code available. A demo binary is available for testing.
Description: A commercial security scanner by the great guys at eeye.

Libnet http://www.packetfactory.net/libnet/

Description: Routines for the construction and handling of network packets. libnet provides a portable framework for low-level network packet writing and handling. . Libnet features portable packet creation interfaces at the IP layer and link layer, as well as a host of supplementary functionality. Still in it's infancy however, the library is evolving quite a bit. Additional functionality and stability are added with each release. . Using libnet, quick and simple packet assembly applications can be whipped up with little effort. With a bit more time, more complex programs can be written (Traceroute and ping were easily rewritten using libnet and libpcap).

Crack / Libcrack http://www.users.dircon.co.uk/~crypto/

Description: Crack 5 is an update version of Alec Muffett's classiclocal password cracker. Traditionally these allowed any user of a system to crack the /etc/passwd and determine the passwords of other users (or root) on the system. Modern systems require you to obtain read access to /etc/shadow in order to perform this. It is still a good idea for sysadmins to run a cracker occasionally to verify that all users have strong passwords.

Cerberus Internet Scanner http://www.cerberus-infosec.co.uk/cis.shtml

Description: CIS is a free security scanner written and maintained by Cerberus Information Security, Ltd and is designed to help administrators locate and fix security holes in their computer systems. Runs on Windows NT or 2000. No source code is provided.

Swatch http://www.stanford.edu/~atkins/swatch/

Description: Swatch was originally written to actively monitor messages as they were written to a log file via the UNIX syslog utility. It has multiple methods of alarming, both visually and by triggering events. The perfect tools for a master loghost. This is a beta release of version 3.0, so please use it with caution. The code is still slightly ahead of the documentation, but examples exist. NOTE: Works flawlessly on Linux (RH5), BSDI and Solaris 2.6 (patched).

OpenBSD http://www.openbsd.org/

Description: The OpenBSD project produces a FREE, multi-platform 4.4BSD-based UNIX-like operating system. Our efforts place emphasis on portability, standardization, correctness, security, and cryptography. OpenBSD supports binary emulation of most programs from SVR4 (Solaris), FreeBSD, Linux, BSDI, SunOS, and HPUX.

Nemesis http://celerity.bartoli.org/nemesis/

Description: The Nemesis Project is designed to be acommandline-based, portable human IP stack for UNIX/Linux. The suite is broken down by protocol, and should allow for useful scripting of injected packet streams from simple shell scripts.

LSOF ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/

Description: List open files. Lsof is a Unix-specific diagnostic tool. Its name stands for LiSt Open Files, and it does just that. It lists information about any files that are open by processes current running on the system. The binary is specific to kernel version 2.2

Lids http://www.turbolinux.com.cn/lids/

Description: The LIDS is an intrusion detection/defense system inLinux kernel. The goal is to protect linux systems against root intrusions, by disabling some system calls in the kernel itself. As you sometimes need to administrate the system, you can disable LIDS protection.

IPTraf http://cebu.mozcom.com/riker/iptraf/

Description: Interactive Colorful IP LAN Monitor IPTraf is an ncurses-based IP LAN monitor that generates various network statistics including TCP info, UDP counts, ICMP and OSPF information, Ethernet load info, node stats, IP checksum errors, and others. . Note that since 2.0.0 IPTraf requires a kernel >= 2.2

IPLog http://ojnk.sourceforge.net/

Description: iplog is a TCP/IP traffic logger. Currently, it is capable of logging TCP, UDP and ICMP traffic. iplog 2.0 is a complete re-write of iplog 1.x, resulting in greater portability and better performance. iplog 2.0 contains all the features of iplog 1.x as well as several new ones. Major new features include a packet filter and detection of more scans and attacks. It currently runs on Linux, FreeBSD, OpenBSD, BSDI and Solaris. Ports to other systems, as well as any contributions at all, are welcome at this time.

Fragrouter http://www.anzen.com/research/nidsbench/

Description: Fragrouter is aimed at testing the correctness of a NIDS,according to the specific TCP/IP attacks listed in the Secure Networks NIDS evasion paper. [2] Other NIDS evasion toolkits which implement these attacks are in circulation among hackers or publically available, and it is assumed that they are currently being used to bypass NIDSs

Queso http://www.apostols.org/projectz/queso/

Note: A couple of the OS detection tests in Queso were later incorporated into Nmap. A paper we wrote on OS detection is available here.
Description: Guess the operating system of a remote machine by looking in the TCP replies.

GPG/PGP http://www.gnupg.org/
http://www.pgp.com/

Description: The GNU Privacy Guard (GnuPG) is a complete and free replacement for PGP, developed in Europe. Because it does not use IDEA or RSA it can be used without any restrictions. GnuPG is a RFC2440 (OpenPGP) compliant application. PGP is the famous encryption program which helps secure your data from eavesdroppers and other risks.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Anonymous author. (1998). Maximum Security. Indianapolis, IN: Sams.

Arquilla, J. & Ronfeldt, D. (1997). In Athena's Camp, Preparing for Conflict in the Information Age. Santa Monica, CA: RAND.

Arquilla, John. (2000). Wasting assets were discussed in psychological warfare class taken by the author.

Atkins, Steve. (1999). Sam Spade author. Sam Spade help files Version 2.0.

Atkins, D., Buis, P., Hare, C., Kelley, R., Nachenberg, C., Nelson, A., Phillips, P., Ritchey, T., Sheldon, T., Snyder, J. (1997). Internet Security, Professional Reference Second Edition. Indianapolis, IN: New Riders.

Attrition. (2000). Online source for news "from the underground". The site tracks some security related information and provides many charts about the state of security on the WWW. Available online at www.attrition.org

Barlow, John (OCT, 2000). "The Next Economy of Ideas". Wired Magazine, 8.10.

Baker, D., Christey, S., Hill, W., & Mann, D. (SEP 9, 1999). "The Development of a Common Enumeration of Vulnerabilities and Exposures". Second International Workshop on recent Advances in Intrusion Detection. MITRE Corporation, McLean, VA. Available online at www.cve.mitre.org/docs/development_of_CVE.html

Bay network and Nokia research center. (OCT 22, 1999). "The Case for IPv6". Internet Architecture Board, Available online at http://www.ietf.org/ietf/1id-abstracts.txt

Bernstein, R. (2000). Powerpoint slides provided in a network attack course at NPS, Monterey, CA during the months of APR–JUN, 2000. Taken by author.

CERT. (APR 4, 2000). "CERT® Incident Note IN-2000-03". Discusses the 911 Worm and its variants. Available online at http://www.cert.org/incident_notes/IN-2000-03.html

CERT. (2000). Online report of the current numbers of attacks available at www.cert.org

CERT (JUL 26, 2000). "CERT Advisory CA-2000-14 Microsoft Outlook and Outlook Express Cache Bypass Vulnerability". Carnegie Mellon University. Available online at ww.cert.org

CERT (Dec 7, 1999). "Results of the Distributed-Systems Intruder Tools Workshop. Pittsburgh, PA. November 2-4, 1999". Carnegie Mellon University. Available online at ww.cert.org

CERT Advisory. (MAY 9, 2000). "CA-2000-04 Love Letter Worm". Available online at www.cert.org/advisories/CA-2000-04.html

CERT/DoS (2000). "Denial Of Service Attacks". From the World Wide Web: http://www.cert.org/tech_tips/denial_of_service.html

CERT/SEI (2000). "The Changing Nature of the Threat". Powerpoint slides. Pittsburgh PA: SEI Carnegie Mellon.

CNET. (FEB 12, 2000) "Hack Leads point to CA universities". Available online at www.cnet.com

Clyman, John. (SEP 1, 2000). "The Constantly Evolving Web is Being Driven by Ever-growing Human Needs". PC Magazine.

Cowan, Crispin., Calton Pu., Dave Maier., Heather Hinton., Jonathan Walpole., Peat Bakke., Steve Beattie., Aaron Grier., Perry Wagle. and Qian Zhang. (2000). "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks". Department of Computer Science and Engineering, Oregon Graduate Institute of Science & Technology. From the World Wide Web: http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard

Denning, D. (1999). Information Warfare and Security. Berkeley, CA: Addison-Wesley.

Dittrich, D. (1999). "Root Kits and hiding files/directories/processes after a break-in. University of Washington". Available on line at http://staff.washington.edu/dittrich/misc/faq/rootkits.faq

Dittrich, D./DoS. (1999). "The DoS Project's "trinoo" distributed denial of service attack tool". University of Washington. Available online at http://staff.washington.edu/dittrich

Dittrich, D. (2000). Extensive DDoS resources and links available on his site. University of Washington. Available online at http://staff.washington.edu/dittrich

Dube, Jonathan. (FEB, 2000). "University computer used in attacks". Available online at www.abcnews.com

Dube, Jonathan. (FEB 25, 2000). "Two More Sites Attacked". Available online at absnews.go.com/sections /teh/dailynews/ webattacks000225.html

Finley, Michelle. (MAY 19, 2000). "Hacker Rails Against New Worm". Wired News. From the World Wide Web: http://www.wired.com

Fyodor (Online alias). (2000). Sources derived from the World Wide Web: http://www.insecure.org

Gartner, John. (APR 25, 2000). "DSL in every 550mhz-plus port". Available online at www.wired.com

Gray, Chris Hables. (1997). Postmodern War; The New Politics of Conflict. New York, New York: The Guilford Press.

Grice, Corey. (February 7, 2000). "How a basic attack crippled Yahoo". Available online at www.cnet.com

Hafner, K., & Lyon, M. (1996). Where Wizards Stay Up Late, The Origins of the Internet. New York, NY: Simon and Shuster.

Hibbard, Justin, (OCT 31, 2000), "Trend number five: wireless", Part of a survey of the ten top trends of 2001, available on line at www.redherring.com

Howard, J. (1997). An Analysis of Security Incidents on the Internet 1989-1995. Thesis paper for Ph.D. at Carnegie Mellon University, Pittsburgh, PA.

Huegen, Craig. (1998) "The Latest In Denial of Service Attacks: "Smurfing" : Description and Information to Minimize Effects". From the World Wide Web: http://www.rsng.net/presentations/nanong11/smurf

Kelsey, Dick. (AUG 17, 2000). "Nielsen: 52 percent of US Homes on Net". Newsbytes. Available online at www.nielsen-netratings.com

Kelsey,Dick (September 11, 2000). "Napster User Count Soars – Media Matrix". Washington D.C. Available online at www.infowar.com/survey/00/survey_091100c_j.shtml

Krebs, Brian. (AUG 28, 2000). "Internet Economy to Employ 10 Million by 2002 – report". Newsbytes. Available online at www.infowar.com

Krebs, Brian. (SEP 11, 2000). "95 Percent Of Public Schools Now Net-Connected Report". Washington D.C. Available online at www.fcc.gov

Kuptz, Jerome. (OCT, 2000). "Independence Array; Gnutella: Unstoppable by Design". Wired Magazine 8.10.

Lemos, Robert. & Mack, Jennifer. (2000). "Web Attacks: FBI launches probe". Available online at www.zdnet.com/ebusiness/stories/0,5918,2435149-2,00.html

Littman, Jonathan. (1996). The Fugitive Game, Online with Kevin Mitnick. New York, NY: Little, Brown and Company.

Loeb, Vernon. (FEB 24, 2000). "Cyberwar's Economic Threat". Washington Post, Washington D.C. pg. 19.

Manjoo, Farhad. (OCT 23, 2000). "Broadband could be Hackland". Available online at www.wired.com

Markoff, J. (FEB10, 2000). "The Strength of the Internet Proves to be its Weakness". The New York Times.

McClure, S., Scambray, J., & Kurtz, G. (1999). Hacking Exposed: Network Security Secrets and Solutions. Berkeley, CA: McGraw-Hill.

McGuire, David. (JUL 10, 2000). "Report: the Internet is, like, really big now". Newsbytes Found online at www.Cyveillance.com

Messmer, Ellen. (FEB 9, 2000). "FBI Opens Investigation to Track Denial-of-Service Attacks". Network World Fusion. Available online at www.nwfusion.com/news/2000/0209fbihack.html

Meunier. P., and Spafford, E. (1999). "Final report of the 2nd Workshop on Research with Security Vulnerability Databases". CERIAS, Purdue University. Available online at citeseer.nj.nec.com/264141.html

Nelson, B., Choi, R., Iacobucci, M., Mitchell, M., Gagnon, G. (1999). Cyberterror, Prospects and Implications. White Paper, Naval Postgraduate School, Monterey, CA.

Netcraft. (2000). Online source for information about the total Internet, including the most widely run servers. Available online at www.netcraft.com

Netice Corporation. (2000). "Advice: Underground: Hacking: Methods: Technical: Buffer Overflow". From the World Wide Web: http://www.netice.com

Northcutt, Stephen. (1999). Network Intrusion Detection, An Analyst's Handbook. Indianapolis, IN: New Riders.

Office of General Counsel. (MAY 1999). "An Assessment of International Legal Issues in Information Operations". Department of Defense.

Patrizo, Andy. (JAN 7, 2000). "SETI Upgrades ET Search". Available online at www.wired.com

PCWorld Online. (2000). "Internet Tools". From the World Wide Web: http://www.pcworld.com/fileworld/file_description/frameset/0,1458,4709,00.html

Pearlman, Kezia. (2000). "Will the Rash of Denial of Service Hacks Hurt tech Stocks?" Available online at www.techtv.com/moneymachine/investing/story/0,3666,2435612,00.html

Pinkerton, J. (MAY 8, 2000). "It's time for Uncle Sam to Foil a Cyber Pearl Harbor". San Francisco Chronicle.

Plummer, Anne. (MAY 12, 2000). "DOD Investigating How "Love Bug" Got Into Classified Systems". Defense Information and Electronics Report. Available online at ebird.dtic.mil

Powell, Dennis E. (2000). "The Real Lessons of ILOVEYOU, Just Wait Until More Sophisticated Scripts Head Our Way". Available online at www.internet.com

Reuters. (JUL 26, 2000). "The Net's Achilles Heel". Available online at www.wired.com/news/print/0,1294,37806,00.html

RFP (online alias). (1999). "Purgatory 101: Learning to cope with the SYNs of the Internet". From the World Wide Web: http://packetstorm.securify.com/papers/contest/RFP.doc

Roblimo. (2000). Online interview posted 16 February, "Security expert Dave Dittrich on DDoS Attacks". Available online at http://slashdot.org/interviews/00/02/16/1836215.shtml

Sans Institute. (2000). "Help Defeat Denial of Service Attacks: Step-by-Step". From the World Wide Web: http://www.sans.org/dosstep/index.htm

Sans Institute. (2000). "How to Eliminate the Ten Most Critical Internet Security Threats; The experts' consensus". JUN 2, 2000. http://www.sans.org

SecurityFocus. (2000). SecurityFocus is a Internet security related website. The chart was derived by SecurityFocus using numbers obtained from BUQTRAQ. Available online at www.securityfocus.com

Seffers, George. (NOV 6, 2000). "DOD Database to Battle Cybercrime". Federal Computer Week. Available online at ebird.dtic.mil/nov2000/s20001113database.htm

Sheppard, Nat. (JUN, 2000). "Technology: Arming Against Cyber Terrorists". Emerge Magazine.

Stallings, W. (2000). Data & Computer Communications, Sixth Edition. Upper Saddle River, NJ: Prentice Hall.

Sullivan, B. (NOV 2, 1999). "Remembering the net crash of '88". Available online at http://www.msnbc.com/news/209745.asp?cp1=1

Thomas, Pierre. (JUN 9, 2000). "FBI Probes report of widespread hacker infiltration". Associated Press story able to found online at www.cnn.com/2000/tech/computing/06/09/hacker.attack.03/index.html

Todd, Bennet. (FEB 2000). "Distributed Denial of Service Attacks". From the World WideWeb: http://www.opensourcefirewall.com/ddos_whitepaper_copy.html

Ullman, Ellen (1998). "The Dumbing-down of Programming". http://www.salonmagazine.com/21st/feature/1998/05/cov_12feature.html

PERSONAL INTERVIEWS

Arquilla, John. & Shimeall, Tim. (MAR 9, 2000). Conversation in Pittsburgh, PA at CERT/SEI discussing possible scenarios including the first hypothetical case in this chapter.

Carpenter, J. Interview conducted on MAR 9, 2000. Mr. Carpenter is a member of the Software Engineering Institute at Carnegie Mellon.

Fithen, W. Interview conducted on MAR 9, 2000. Mr. Fithen is a member of the Software Engineering Institute at Carnegie Mellon.

Konda, S. Ph.D. Interview conducted on MAR 8, 2000. Mr. Konda is a member of the Software Engineering Institute at Carnegie Mellon.

Longstaff, T. Ph.D. Interview conducted on MAR 9, 2000. Mr. Longstaff is a member of the Software Engineering Institute at Carnegie Mellon.

McHugh, J. Ph.D. Interview conducted on MAR 6, 2000. Mr. McHugh is a member of the Software Engineering Institute at Carnegie Mellon.

Shimeall, T. Interview conducted on MAR 9, 2000. Mr. Shimeall is a member of the Software Engineering Institute at Carnegie Mellon.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..2
   8725 John J. Kingman Rd. Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library..2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943

3. Professor Arquilla..1
   (Code SO/Ar)
   Naval Postgraduate School
   Monterey, CA 93943

4. Timothy J. Shimeall...1
   Carnegie Mellon/Software Engineering Institute
   Pittsburgh, PA 15213-3890

5. The Honorable Brian Sheridan...1
   Assistant Secretary of Defense for SO/LIC
   The Pentagon, RM 2E258
   Washington, DC 20301-2500

6. GEN Charles Holland..1
   Commander in Chief
   US Special Operations Command
   MacDill AFB, FL 33608-6001

7. RADM Eric T. Olson...1
   Commander
   Naval Special Warfare Command
   NAB Coronado
   San Diego, CA 92155

8. MG Dell Dailey...1
   Commander
   Joint Special Operations Command
   Ft. Bragg, NC 29307

9. United States Special Operations Command...2
   SOOP-JE
   7701 Tampa Point Blvd
   McDill AFB, FL 33621-5323

10. COL Donn Kegel...1
National War College
300 D Street
Fort McNair
Washington, DC 20319-5078

11. Jennifer Duncan...8
Special Operations Academic Group
Code (CC/Jd)
Naval Postgraduate School
Monterey, CA 93943-5000

12. Library...1
Army War College
Carlisle Barracks, PA 17013

13. Library...1
Naval War College
Newport, RI 02840

14. Strategic Studies Group (SSG)...1
Naval War College
Newport, RI 02840

15. Department of Military Strategy...1
National War College (NWMS)
Ft. Leslie J. McNair
Washington, DC 20319-6111

16. US Army Command and General Staff College...1
ATTN: Library
Ft. Leavenworth, KS 66027-6900

17. Library...1
Air War College
Maxwell AFB, AL 36112-6428

18. US Military Academy...1
ATTN: Library
West Point, NY 10996

19. US Naval Academy...1
ATTN: Library
Annapolis, MD 21412

20. Commander..1
Naval Special Warfare Group One
NAB Coronado
San Diego, CA 92155

21. Commander..1
Naval Special Warfare Center
NAB Coronado
San Diego, CA 92155

22. US Air Force Special Operations School..1
EDO, Alison Bldg, 357 Tully St.
Hurlburt Fld FL 32544-5800

23. Michael D. Stull..1
39512 Platero Place
Fremont, CA 94539

24. Craig Robinson..1
Defense Intelligence Agency
200 McDill Blvd
Building 6000, Room A4-908
Washington, DC 20340-51000